SIGS

1999's Cumulative Author Index

December 1999
Volume 4, Number 12
$4.95 US/$7 Can/$4.95 Int'l

Java™ REPORT

THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

THE CLIENT SERVER

Build Maintainable Systems Quickly With JSPs

A Guide Through the More Advanced Features of the JavaBean Spec Using a Hierarchical Graphical Tree

Solving UI Problems By Extending an AWT and a Swing Widget

iBus—A Publish-and-Subscribe Messaging Middleware

What Makes a Successful Java Interface?

Building Better E-Mail With JavaScript

The NULL OBJECT—
A Useful, Tactical Pattern

REVIEWED
ObjectSpace's Voyager ORB 3.0

IPD
0 74470 88941 0 12

Get a Jump Start on Securing Your e-Business Applications.

Connect to the power of RSA BSAFE® e-security components.

It seems like the whole world is out there racing to record e-business profits. So where does that leave you? If you find yourself stuck curbside, while competitors leave you behind to sort through e-security issues, then RSA Security has the power boost you need – whether you develop apps for Web business, general client-server, remote management or embedded devices.

RSA BSAFE components. Mega volts of security. Our RSA BSAFE e-security components give you all the tools you need to power up your e-business applications. For SSL in C, SSL for Java™ technology and S/MIME, we get you rolling in a way those under-powered off-the-shelf browsers just can't match. In one jolt, you'll get full-strength crypto (available worldwide*), more extensive sample code and protocol links for services such as Telnet, FTP, NNTP, SMTP and POP3. And, by combining SSL and S/MIME security protocols with the industry's most trusted cryptography, RSA BSAFE security gives you the jump you need to get to market faster.

RSA BSAFE e-security. Power you control. Don't be stalled by e-security issues. With RSA BSAFE security components you're in charge of the implementation of our airtight security within your application. You control the interface, error recovery and the best balance of strength vs. accessibility for your needs. Best of all, you start off with the confidence that your e-security operates independently of changes to a browser. Your apps will always be completely interoperable.

RSA Security. Rev up with the best. Connect with RSA Security and you work with the industry leader. We've been the driving force behind putting SSL and S/MIME on the map as industry standard security protocols. No one knows more about how to build secure applications, so you can count on us to provide tools and support you can use to meet your goals on time and on budget.

Get a jump start on the competition. **Qualify today for a free RSA BSAFE SSL-C, SSL-J or S/MIME-C software development kit* and get charged up.**

1.800.782.5453 or www.rsasecurity.com/go/jumpstart

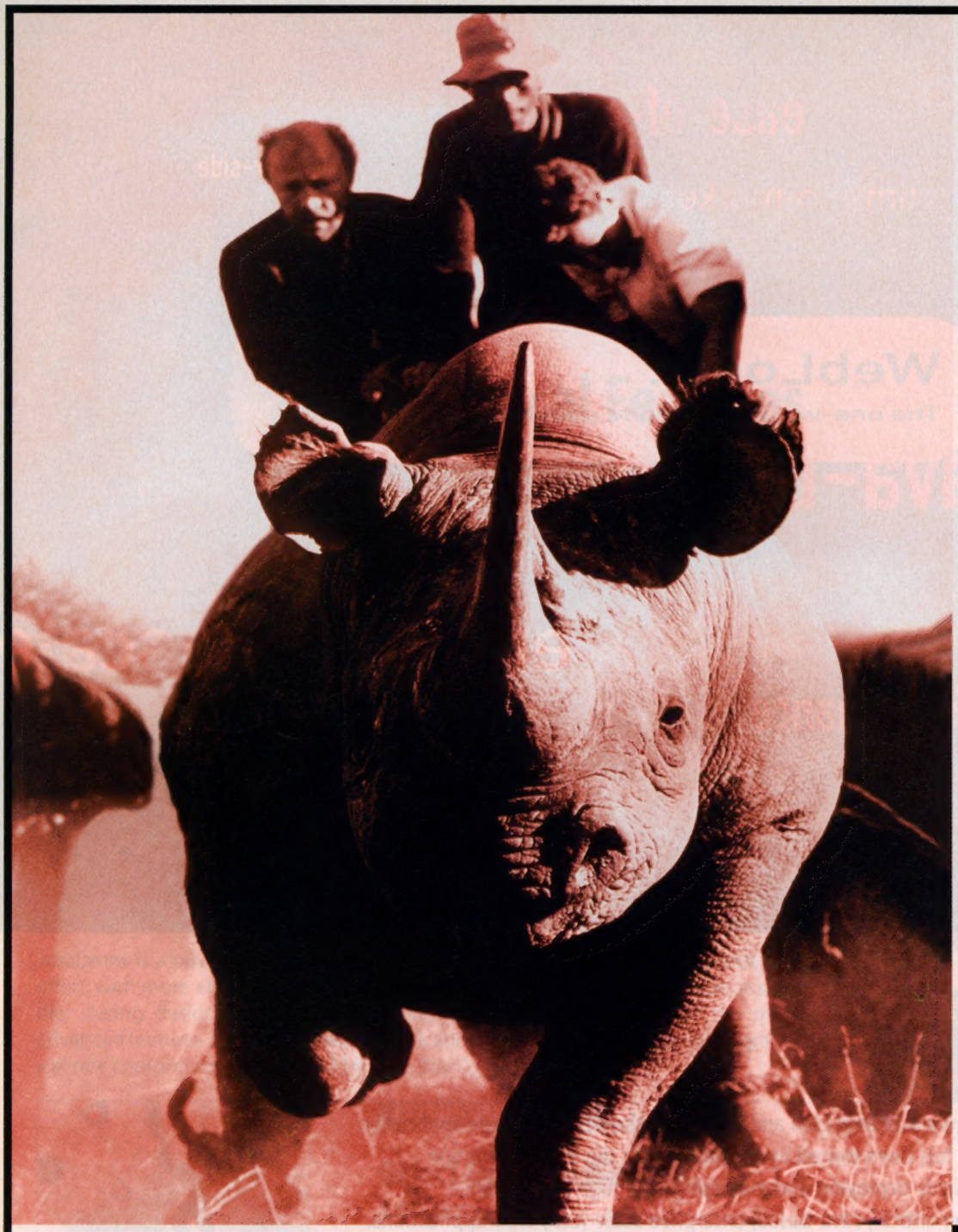
RSA SECURITY™

The Most Trusted Name in e-Security

RSA SECURED™

Products displaying this symbol contain the most trusted e-security technology

*Export or other restrictions may apply. RSA, BSAFE, the RSA Security and RSA Secured logos are trademarks or registered trademarks of RSA Security Inc. Java is a trademark of Sun Microsystems, Inc. © 1999 RSA Security Inc. All rights reserved.



UNTIL NOW, JAVA FOR EMBEDDED SYSTEMS HAS BEEN JUST TOO BIG AND UNPREDICTABLE.

INSIGNIA SOLUTIONS

JEODE™

You want the power of Java™. But for your embedded application, it's just too hefty, unpredictable, and not fast enough. Insignia Solutions® introduces Jeode™, our implementation of Java, which is fully compliant with Sun's EmbeddedJava™ specification and uniquely designed for embedded systems. Jeode lets you optimize size, performance and predictability for your specific embedded application. Featuring our proprietary EVM™ (Embedded Virtual Machine™), Jeode is highly configurable and tunable to achieve optimal performance in resource-constrained embedded systems. The EVM provides adaptive optimizing dynamic compilation for fast execution and concurrent garbage collection for predictable behavior. With more than a decade of delivering virtual machines, Insignia Solutions is already providing Jeode to Fortune 500 makers of networking equipment, PDAs, mass storage systems, consumer electronic devices and more. So, can Java be small enough, fast enough and predictable enough for your embedded system? The answer is a resounding yes! With Jeode. Call **1.800.848.7677** or visit **www.insignia.com**.

Code JR

Insignia Solutions is a registered trademark and Jene, EVM, and Embedded Virtual Machine are trademarks of Insignia Solutions, Inc. Sun, Java, and EmbeddedJava are trademarks or registered trademarks of Sun Microsystems, Inc. All other trademarks are the property of their respective holders.

ease-of-use

leading-edge

server-side

time-to-market

WebLogic.

The one-word solution to your hyphenated needs.

java-based

mission-critical

best-of-breed

enterprise-scale

high-performance

time-to-market

Are Web applications part of your IT strategy?

Go to www.beasys.com and learn why the BEA WebLogic application server has been touted by the industry as the leading solution for quickly deploying and managing your high-performance, mission-critical, Java-based, enterprise-scale (etc.) Web applications.

Java Report 4 Cup Rating

BEA WebLogic®

www.beasys.com

© 1999 BEA Systems, Inc. All rights reserved. BEA, the BEA logo, and WebLogic are registered trademarks, and the BEA logo is a trademark of BEA Systems, Inc. Java is a trademark or registered trademark of Sun Microsystems, Inc. Other trademarks are the property of their respective owners.

bea™

Java™ REPORT

THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

December 1999
Vol. 4, No. 12

Technical Features

ENTERPRISE JAVA

Issues in Java Server Pages and Servlet Design 24

John Akerley, Alex Koutsoumbos, Murtuza Hashim, and Angelo Maffione

Java Server Pages are a new technology and the tools and techniques are evolving. Naming the patterns and paradigms possible in Web applications will help you formulate design ideas and recognize these patterns and paradigms in other Web applications. Here are some design issues uncovered while building an Internet banking system at the San Jose International Technical Support Organization at IBM.

VISUAL JAVA

Advanced JavaBeans Development: Enhancing a Hierarchical Graphical Tree Component 32

Eldon Metz

JavaBeans is a powerful component model. You can build a simple Bean with little effort; with more effort, you can add features that significantly aid the Bean assembler during design. Metz guides us through the implementation of the more advanced features of the JavaBeans specification by enhancing a hierarchical graphical tree component represented by class TreeControl.

COMPONENT JAVA

Object-Oriented Widgets 44

Robb Shecter

Solving UI problems by extending an AWT and a Swing Widget to make them act more object-oriented. The ability to directly display real objects in UI widgets encourages better more flexible designs.

ENTERPRISE JAVA

iBus—A Software Bus Middleware for the Java Platform 54

Marcel Altherr, Martin Erzberger, and Silvano Maffeis

Businesses need to react to important events immediately and to disseminate such events to many receivers. This calls for a new breed of Middleware that is event-oriented and adaptable to various transport mechanisms and quality-of-service requirements. Our goal has been to provide this type of Middleware by developing the iBus—a publish-and-subscribe messaging Middleware that can run atop various IP and non-IP communication protocols.

Columns

MODELING AND JAVA 64

The Presentation and Domain layers

Craig Larman

Craig explores the design of the presentation and application logic layers and the relationship between them. He re-examines real use cases, the coupling and communication between them and considers the question, "What's in a Name?"

JAVASCRIPTING 70

Building better email with JavaScript

Steven W. Disbrow

Is it possible to create a Web page that can email the contents of a form and then redirect the client to a "thank you" page? The catch is, it all has to be done on the client side, with no server-side utilities involved, and, most important, the resulting emailed data has to look good.

PATTERNS IN JAVA 74

Something for nothing

Kevlin Henney

Kevlin looks beyond the relatively familiar GoF territory, presenting examples that demonstrate patterns of use to the Java developer, as well as some revisits of GoF patterns from a different perspective. This month, we discuss the NULL OBJECT, a useful tactical pattern perhaps best described as something based on nothing.

SCOTT'S SOLUTIONS 88

Creating good Java interfaces

Scott Oaks

Scott does something a little different this month—he answers the question: What makes a good interface in Java? There's some rationale here, but none of the "it works this way" sort of answers Scott usually attempts. It's a critical question, especially when writing good object-oriented code that can be easily shared among developers—whether they're coworkers down the hall or anywhere around the world!

Departments

EDITOR'S NOTE 6

The best things in life are free

Dwight Deugo

BUZZ 10

Philip J. Gill

Embedded Java Pace Picks Up; IBM Formally Debuts developerWorks Site; Optimizeit Bundled; Microware Aims at MP3 Devices; and more.

PRODUCT REVIEW 14

Achieving Synergy: Voyager ORB 3.0 From ObjectSpace

Juancarlo Añez

A great piece of software that anyone developing distributed clients will want to check out. It implements every imaginable distributed computing feature with only a few abstractions and a small and easy-to-use API. No other product on the market makes client programming easier!

AD INDEX 83

THE 1999 AUTHOR INDEX 84

Check out Java Report Online!

- Selected features and columns from the print edition
- All code listings are available online!
- 1996–1999 Cumulative Author Index
- Java Report Archives

http://www.javareport.com

Cover illustration by Credit

TEST DRIVE

Persistence PowerTier 5 for EJB

Download FREE Evaluation Software!

www.persistence.com/Download.html Priority Code JR10

The Engine for E-Commerce

Developing high-volume, high performance e-commerce applications?

Download a free evaluation copy of PowerTier™ 5 for Enterprise JavaBeans™ and discover how Persistence brings your e-commerce system to the next level of performance, scalability and personalization!

PowerTier 5 for EJB

- *Delivers application performance through its in-memory cache*
- *Ensures scalability and high availability with PowerSync technology*
- *Reduces time-to-market with patented O-R mapping*

Persistence

The Engine for E-Commerce

800.803.8491 • info@persistence.com • www.persistence.com

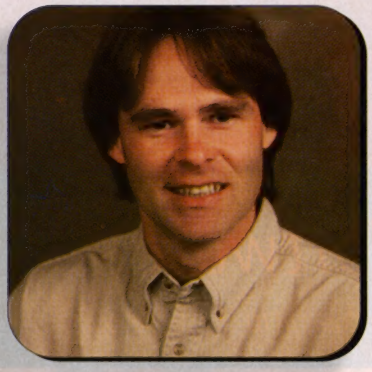
© 1999 Persistence Software, Inc. All Rights Reserved. Persistence is a registered trademark and PowerTier is a trademark of Persistence Software in the United States and other countries. U.S. Patent Nos. 5,499,371; 5,615,362; 5,706,506

Editor's Note

Dwight Deugo / deugo@scs.carleton.ca

The best things in life are free

I GUESS SUN and Microsoft are never going to kiss and make up. Why? I think it's because their business models are in direct conflict with one another. Sun's Java software model seems simple enough: Give as much as you can away for free. There may be a law about dumping your product for a ridiculously low price in a given market, but no one said you can't give it away. After all, Sun is not making any profit following this approach. Yeah, that's the ticket. And, I also seem to remember something about the tobacco manufacturers giving away cigarettes to soldiers during the second World War. They never made any profit from that either. Well, at least not in the beginning. So how does Sun make any money? Don't forget, Sun sells hardware, services, and support too—you must have something to run all that free software on and someone to tell you how to do it.



The birth and subsequent freely downloadable version of Java is undeniable proof of this model. In the end, though, I don't think Java had as much of an impact on Microsoft as many anticipated. It may have caused a few bruises, but I don't think anyone who has owned Microsoft stock for the last few years has had much to complain about. However, if it worked once, why not try it again? And that's what Sun has done.

Recently, Sun announced its StarPortal initiative to provide word processing, presentation graphics, spreadsheet, and other office software tools to any Web browser, and eventually to portable devices, all written in Java. Companies such as Corel, Lotus, and IBM have tried to do this before, with only limited success, but Sun doesn't seem too deterred by this and is already out of the gate and running. As a prelude, Sun is making StarOffice, a fully featured office productivity suite, freely available now. StarOffice provides word processing, a spreadsheet, graphic design, presentations, database access, an HTML editor, a mail/news reader, an event planner, and even a formula editor, in only 65 MB.

Now what would happen if Microsoft decided to give its office productivity suite away free? And then Windows 95, 98, and finally, NT?

Well there is always the service and support business to fall back on.

While it's true the best things in life are free, I would still be willing to pay good money for something that works well. However, all things considered equal, I guess I'll just have to keep my money in my pocket and wait to see what comes along.

Dwight Deugo

Java™ Report (ISSN 1086-4660) is published monthly by 101communications LLC, 71 West 23rd St., 7th Floor, New York, NY 10010, 212.242.7447. © Copyright 1999 by 101communications, All rights reserved. Periodicals postage paid at New York, NY, and additional mailing offices. Regular subscription rates—$39 domestic, $64 Canada and Mexico, $88 foreign; institutional—$139 domestic, $164 Canada and Mexico, $188 foreign. Postmaster: Send address changes and subscription orders to: Java Report, P.O. Box 5050, Brentwood, TN 37024-5050. GST (Canada) #R895884229. For service on current subscriptions: in North America call 800.361.1279 (v) or 615.370.4845 (f); (custserv@cwixmail.com (email); outside North America +44(0)1858 435 302 (v), +44(0) 1858 434 958 (f); custserv@cwixmail.com (email). Statements of opinion and fact are made on the responsibility of the authors alone and do not imply an opinion on the part of 101communications, or the editorial staff. All trademarks are the property of their respective owners. Printed in the USA.

Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other countries. *Java™ Report* is independent of Sun Microsystems Inc.

Java™ REPORT

THE INDEPENDENT SOURCE FOR JAVA DEVELOPME

BPA INTERNATIONAL

Editor-in-Chief
Dwight Deugo; *deugo@scs.carleton.ca*

Executive Editor/Managing Editor
Kathleen M. Major; *kmajor@sigs.com*

Art Director
Emily Rich; *erich@101communications.com*

Assistant Managing Editor/ Associate Online Editor
Dan Olawski; *dolawski@sigs.com*

Assistant Editor
Melissa Shepherd; *mshepherd@sigs.com*

Associate Art Director
Fernando Quiñones; *fquinones@sigs.com*

Associate Editor
Philip J. Gill; *philipgill@aol.com*

Product Editor
John K. Waters; *john@watersworks.com*

Intern
Tom DeSimone; *tdesimone@sigs.com*

COLUMNISTS
Steve Ball, *Independent Consultant*
John Crupi, *Sun Microsystems*
Mark Davis, *IBM*
Steven Disbrow, *EGO Systems*
David S. Frankel, *Genesis Development Corp.*
Vincent Hardy, *Sun Microsystems*
David Jordan, *Ericsson*
Lowell Kaplan, *Random Walk Computing*
Craig Larman, *Object Space*
Dan Malks, *Sun Microsystems*
John Miller Crawford, *Dave Clark Design Associates*
Scott Oaks, *Sun Microsystems*
Govind Seshadri, *Strategic Technology Resources*
Tig Tillinghast, *Internet Profiles*

101 communications SIGS PUBLICATIONS

101communications LLC
Curt Hessler, *Chairman and CEO*
Jeffrey S. Klein, *President and COO*
Michael Weaver, Senior *Vice President and CFO*
Christine Oldenbrook, *Vice President/Circulation*
Charles P. Shively, *Group Publisher*

Corporate Headquarters
9121 Oakdale Ave., Ste. 101, Chatsworth, CA 91311
v. 818.734.1520 • www.101communications.com

For subscription customer service please call toll-free 800.361.1279

ADVISORY BOARD
François Bancilhon, *Ardent Technology*
Grady Booch, *Rational Software Corp.*
Alan Brown, *Sterling Software*
Thomas Keffer, *Rogue Wave Software*
Bertrand Meyer, *ISE*
Meilir Page-Jones, *Wayland Systems*
Jan Popkin, *Popkin Software and Systems, Inc.*
Kenneth S. Rubin, *Salient*

Publishers of *JOOP, C++ Report, Java™ Report, Application Development Trends, OBJEKTSpektrum (Germany), Java™SPEKTRUM (Germany), Application Development Advisor (UK), Java™ Report Online, C++ Report Online,* and *JOOP Online.*

500,000 downloads of Sun Microsystems' Java™ Foundation Classes.

Multiple platforms.

One installer.

InstallAnywhere®

ENTERPRISE EDITION

www.ZeroG.com

Deploying enterprise software across multiple platforms? Whether they're applications, libraries, or servlets written in Java or platform specific code, InstallAnywhere is the only solution that creates a single, universal installer that can deploy software from the Internet or CD-ROM to virtually any platform, including Windows, Solaris, Linux, AIX, and Mac OS. Try it today and find out why companies such as Sun Microsystems, Adobe and FedEx are using InstallAnywhere to ease the deployment of their software across multiple platforms.

Any application. Any platform. The only solution.

© 1999 Zero G Software, Inc. InstallAnywhere, Zero G Software, the Zero G logo, and ZeroG.com are trademarks or registered trademarks of Zero G Software, Inc. Sun, Sun Microsystems, Java, Java Foundation Classes, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All other trademarks and registered trademarks are the property of their respective holders.

Bipolar CORBA Objects in Java

I enjoyed your article, "How to Create Bipolar CORBA Objects in Java," in the recent edition of *Java Report* (Mak, R., Sept. 1999, Vol. 4, No. 9). I was, however, disappointed that the article did not provide more information on the limitations of the approach so that the readers could make informed decisions about where and when the techniques can be used.

I believe that the testing approach you have suggested is fine for two-way CORBA calls, but when the client is making one-way calls and expecting a callback, the bipolar approach has significantly different characteristics than when running in a client/server environment. When the client is making one-way calls and expecting an asynchronous callback from the server, the client's stack does not have the methods leading up to the method invoking the method on the server. These methods may have code in them, which expects that the response has not been received. In the bipolar approach, after making the server request, the response is processed and then the method invoking the server request is completed and all the methods in the call stack are processed.

This can have drastically different results if the code is not designed for this to happen.

I believe that the approach mentioned is very useful, but that one should have an understanding of its limitations so that test beds can be developed that are not prone to the limitations. Only by presenting both the advantages and disadvantages of an approach can the true value of the approach be determined.

Chris Atkins
atkicd@aud.alcatel.com

Reply

Thank you very much for reading my article, and especially for your comments. The bipolar method will work well under the conditions that I described. Under other conditions, such as the asynchronous situation that you mentioned, the method may not work as expected. You are absolutely correct in pointing out that for any debugging technique, programmers should be aware of its limitations.

Ron Mak

Faulty Code?

In the article, "Distributed Event Notification Using RMI," (Landis, S., *Java Report*, July 1999, Vol. 4, No. 7) the code in Listing 1 will not work as expected: The EventService.subscribe() method attempts to prevent duplicate subscriptions by first constructing the Subscription object and then passing it as an input to Vector.contains() method. The problem of this approach is that the Vector will be searched using Object.equals() method, which compares objects' references, not the contents of their fields. To fix this problem, the Subscription class needs to override the Object.equals() method to do field-by-field comparison. The Vector.removeElement() call made from the EventService.unsubscribe() method won't work for the same reason. This code seems to have never been tested.

Vadim Spector
vspector@ispchannel.com

Reply

You are correct to point out that Object.equals() must be overridden. The goal of the article was to demonstrate how easy it is to extend an existing in-process pattern—as described in "Event Notifier: A Pattern for Event Notification" (Gupta, S., J. M. Hartkopf, and S. Ramaswamy, Java Report, *July 1998, Vol. 3, No. 7), to a distributed pattern using RMI. To that end, I modified as little of the original code as possible. Thanks for pointing out the problem.*

Sean Landis

Variations On Creating Bipolar CORBA Objects

I read your article, "How to Create Bipolar CORBA Objects in Java," (Mak, R., *Java Report*, Sept. 1999, Vol. 4, No. 9) and I have to say I am a little bit confused about the way to create the server object locally.

I am using a different ORB-implementation as the author's and its syntax (name of the automatically generated classes etc.) is different. To create a local server object in the ClientApp.java code I would just use (this is the syntax of the author's ORB):

```
helloObject = new _tie_Hello(new HelloImpl());
instead of
helloObject = new LocalHello();
```

Why should I create a new class LocalHello at all? Why stub out org.com.omg.CORBA.Object methods, which never get called, if you can inherit already implemented methods from _HelloImplBase? Why create the standard constructor? Why not use the generated class _tie_Hello? Why think about a tool to support the generation of the new class?

Furthermore, I am using the same scheme for C++ and it works the same way. The only thing I have to care about is reference counting. The way the ORB I use is designed, (in C++) I shouldn't use:

```
Hello *helloObject = new Hello_skel_tie<Hello_impl_tie>(
  new Hello_impl_tie);
```

and later:

```
delete helloObject; // error!
```

but instead:

```
Hello_var *helloObject = new Hello_skel_tie<Hello_impl_tie>(
  new Hello_impl_tie);
```

and the object goes out of scope automatically.

I am interested in your answer but would probably be surprised if your ORB would not allow this more painless way to create server objects locally.

Udo Hamacher
udo.hamacher@dfs.com

WRITE TO US!!

We'd like to hear from you!
Send comments, suggestions for topics, etc., to
Kathleen Major at kmajor@sigs.com

IBM

DB2 Universal Database™ delivers "right now" response for a world of users on platforms from Sun™ Solaris® to Linux to Windows NT.® It's the most powerful customer-service engine on the Web. What's behind your e-business? Try DB2® free at www.ibm.com/software/soul/db2

Software is the soul of e-business.™

IBM, the e-business logo, DB2, DB2 Universal Database and Software is the soul of e-business. are trademarks of International Business Machines Corporation in the United States and/or other countries. Sun and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Windows NT is a trademark of Microsoft Corporation in the United States, other countries, or both. ©1999 IBM Corp. All rights reserved.

IBM Formally Debuts developerWorks Site

After a three to four month beta release, **IBM (www.ibm.com)** has officially debuted the developerWorks **(www.ibm.com /developer/)** Web portal for developers.

The site contains seven "zones," or individual forums, based on specific technologies. These include forums on Java, XML, Open Source, UniCode, Web Architectures, Security, and Linux. Each site contains information and source code that is available for free download. In the Open Source Forum, developers can also participate in the development and enhancement of existing technologies, following the open source model of the Apache Web server, said **Chris Bahr**, IBM's program director for the site.

Of interest to Java developers in the Open Source Forum is Jikes, IBM's Java Just-in-Time (JIT) compiler. The site also includes ICU, a set of classes for UniCode, and should shortly also include Data Explorer, a data visualization technology.

In addition, the Java Zone has more than 360,000 unique instances of Java code, class libraries, and technical information, said Bahr. It also includes IBM's jCentral search engine, which catalogs Java sites, source code, classes, and other materials available to developers over the Web.

jCentral has been augmented with xCentral, a version of the search engine for XML sites and technologies that are available over the Web, notes **Dirk Nicol**, developerWorks' development manager. "We used the same technology in the jCentral search engine to provide developers a catalog to access XML and DXML source code sites on the Web," said Nicol.

While IBM will get no direct and immediate benefit from developerWorks said Bahr. IBM believes that by fostering these technologies, it can indirectly benefit from an expanding marketplace.

Optimizeit Bundled

A two-week trial demo CD of Optimizeit 3.0 Professional Java performance tool is shipping with the new iPlanet Web Server EE 4.0 package

continued on opposite page ➤➤

Embedded Java Pace Picks Up

Robert Pizzo Images.com

The first signs of fall may have been in the air outside the San Jose, CA, Convention Center the last week of September, but inside, competition between competing visions and versions of Java technology was heating up. Embedded Java rivals **Sun Microsystems Inc. (www.sun.com) and Hewlett-Packard Co. (HP; www.hp.com)** both sought to highlight their efforts in embedded Java and to spark developer interest and win them to their camps.

Sun promoted its three-pronged strategy for Java in embedded systems: Java technology tailored for the embedded market in the form of the Java2 Platform Micro Edition (J2ME), recruiting industry support for bringing Java technology to real-time systems, and providing engineering services to help developers put that technology to work.

The focus of Sun's efforts is the J2ME, intended for both the consumer and embedded markets. As part of the J2ME program, Sun is working with device manufacturers to develop various "profiles" for the different markets and devices the J2ME will support. These profiles consist of a set of APIs optimized for different kinds of products and a Java Virtual Machine (JVM) appropriate for the features and memory footprint of those devices.

Profiles currently under development include the Personal/Web Profile, which includes APIs for Web-connected consumer devices, and the Mobile Information Device (wireless) Profile, which includes specific APIs for smartphones and other devices. Manufacturer representatives from other potential markets, including point-of-sale (POS) terminals, handheld communicators, digital TV, and the automotive industry, are also participating in the community process to develop additional profiles for J2ME, Sun said.

Sun also released the first draft for review of the Real-Time Specification for Java that it has been developing through Sun's Community Process. Sun said the specification will provide device manufacturers the ability to meet deterministic, time-critical constraints necessary for their Java technology-based applications.

Meanwhile, Sun said its new Engineering Services Program, which provides technical resources and expertise focused on allowing customers to reduce development costs, speeds time-to-market, creates better products, and builds devices to meet industry standards.

Not to be left out, HP announced the Chai Appliance Platform, which it describes as [an] "integrated suite of software products designed to power the next generation of memory-constrained intelligent appliances."

The Chai Appliance Platform incorporates previously released embedded-software products, such as ChaiVM, a scal-

able virtual machine (VM); the Chai Server, an embedded Web-based application server; ChaiFreezeDry, a fast Chai compiler; ChaiAWT, an Abstract Windowing Toolkit for creating graphical displays for devices; ChaiAppliance Plug-and-Play, a Jini rival for spontaneous network connectivity; and TurboChai, an "ahead-of-time" compiler.

HP also introduced three new products to the family at the embedded systems conference. These include: Chai/e-speak, which enables intelligent appliances to participate in e-services; Chai/OpenView, which enables devices to be managed by HP OpenView Network Node Manager; and ChaiFarer, an embedded XML Web browser.

HP also released a new ChaiVM, version 4.0. HP said ChaiVM 4.0 increases functionality through the addition of new Java class libraries, including beans, text, sql, security, and math libraries. ChaiVM 4.0 also contains new hard real-time performance improvements that enable real-time Java applications to execute with deterministic response times of less than 125 microseconds.

Finally, HP said three companies—**Hitachi Ltd. (www.hitachi.com)**, HP's own Thin Client Operation, and Siemens AG **(www.siemens.de/ic/en/default.htm)**—are using the Chai Appliance Platform in their products now. These products, it adds, are expected to ship before year's end.

Microware Aims at MP3 Devices

Hoping to capitalize on the growing popularity of MP3, **Microware Systems Corp. (www.microware.com)** used the Embedded Systems Conference '99 to introduce a new version of its OS-9 real-time operating system (RTOS) that it said is the first to embed MP3 audio capabilities.

"The Internet audio market is growing at a phenomenal pace," said **Bob Egner**, marketing director at Microware. "It has been estimated that 1.5 million MP3 audio players will be sold this year, 3 million next year, and 7.8 million in 2001. By 2005, industry analysts predict 30 million MP3 players will be sold, making them more popular than cassette players. Developers who are currently working on advanced designs and building MP3 into embedded devices, now have a fast and robust MP3 engine available through Microware's OS-9 technology."

Microware, based in Des Moines, IA, has combined its OS-9 RTOS with Java technologies, the Kalos development tools, and Internet Application Suite from **Espial Group Inc. (www.espial.com)**, San Jose, CA, and the **Xaudio (www.xaudio.com)** MPEG technology. Xaudio, developed by **MpegTV** in San Francisco is currently being licensed for use in audio devices, video games, and Internet audio devices.

Microware said this combination should provide a complete software solution for developers in the fast-growing embedded Internet-enabled audio device market, particularly MP3 audio-capable devices.

Flashline.com Expands

Flashline.com (www.flashline.com) an online digital marketplace for reusable software components, has revamped its online software components retail site.

To begin with, the site is now called Components-by-Design **(www.flashline.com/components/developers/)**, to reflect its widening array of other software components. Formerly the site was known as Beans-by-Design, and sold only JavaBeans software components.

Among the new features, the site includes support for COM and CORBA components, JavaBeans, and Enterprise JavaBeans (EJB) components.

In addition, Flashline.com has altered its submission process for companies who want to contract out the development of custom software components. Now, users interested in having developers develop new Java, COM, or CORBA components post their requests to the Components-by-Design site, then allow a time for comments from developers to refine their request, before actually putting it out for contract.

Founder and president **Charles Stack** said the change in process was necessary because developers who bid on requests

➤➤ *continued from previous page*
from the Sun–Netscape Alliance.

Optimizeit, from **Intuitive Systems Inc. (www.optimizeit.com)**, Cupertino, CA, is a Java profiling tool that allows developers to track and resolve performance problems in Java programs.

With Optimizeit, Java developers can be sure that the high performance they get from their iPlanet Web Server is not slowed down due to performance bottlenecks or memory leaks in their Java code.

New Java Center to Open

Doctor Design Inc. (DDI; www.doctordesign.com), the San Diego-based design center of Silicon Valley's **Integrated Systems Inc. (ISI; www.isi.com)** has entered into an agreement with **Sun Microsystems Inc. (www.sun.com)** to become a "center of excellence" for the development and design of Java technology-based embedded products.

Under the agreement, Sun will provide technical and specifications support for DDI's internal efforts to port the J2ME, a version of the Java platform optimized for embedded, function-specific devices. DDI will also optimize the technology for an emerging class of intelligent Internet-connected appliances. The ongoing development effort will also integrate Jini connection technology.

Working together, the two companies said they hope to generate up to 100 design wins for new products, which would in turn result in up to 10 million Java technology-based embedded devices reaching the market within the next two years.

Insignia's Jeode Comes to CodeWarrior

Insignia Solutions Inc. (www.insignia.com) of Mountain View, CA, and **Metrowerks Corp. (www.metrowerks.com)** of Austin, TX, have disclosed that Insignia's Jeode platform is now integrated with Metrowerks Integrated Development Environment (IDE). In addition, a CodeWarrior plug-in is now provided by Insignia as part of its standard installation CD, offering Jeode Embedded Virtual Machine (EVM). The Jeode platform is Insignia's implementation of Sun's PersonalJava and EmbeddedJava specifications. It is specifically designed for embedded devices.

Kurt Vargo/Images.com

First EJB 1.1 Servers Expected

Expect a flood of applications servers supporting the EJB 1.1 specification to start hitting the market and the Web before the end of this month. With the EJB 1.1 spec firmed up at JavaOne earlier this year, vendors are now racing to be first out of the gate with production releases.

Among those already delivered in beta form is an open source EJB 1.1 application server from **BullSoft (www.bullsoft.com)**, the software subsidiary of France's Groupe Bull. Called JOnAS, for Java Open App lication Server, this EJB 1.1 application server platform was available for free download from **www.bullsoft.com/ejb** in September. According to BullSoft, JOnAS consists of a Java Transaction Monitor (JTM), a set of Java classes implementing EJB, and a set of tools for generating containers and deployment descriptions.

JOnAS supports stateless and stateful Session Beans; Entity Beans with either Bean Managed Persistence or Container Managed Persistence [both on JDBC; all transaction attributes; the JTA 1.0.1; Remote Method Invocation (RMI); and the Java Naming Directory Interface (JNDI)].

Another early EJB 1.1 app server, a commercial implementation, was also available in beta form in September, from **Iona Technologies (www.iona.ie)**, the Dublin, Ireland-based CORBA software house. It is called the OrbixHome EJB server, and apparently was based on a product Iona bought from another company.

All the major application server vendors—including **IBM Corp. (www.ibm.com), BEA Systems Inc. (www.beasys.com)** and its WebLogic division, and **Oracle Corp. (www.oracle.com)**—have promised to deliver EJB 1.1-compliant products by year's end.

The Theory Center Launches EJB Training Services

Need to come up to speed on building EJBs?

Boston-based **The Theory Center Inc. (www.theorycenter.com)** is offering Launch- EJB Series, a professional services program that includes on-site training, formal classroom education, and consulting services for organizations looking to develop and implement standards-based eBusiness applications.

The program is designed to educate customers on the latest developments in EJB and component technologies, and to provide advanced EJB training for experienced developers. The Theory Center's LaunchEJB Series is targeted to application architects and design staff with EJB knowledge and skill levels ranging from first-time users to advanced developers.

The series includes an intensive five-day introductory training course, Launch EJB Concepts, and three consulting services tracks: LaunchEJB Quickstart, LaunchEJB Proof, and LaunchEJB Immersion.

Sun–Netscape Alliance Ships All-Java iPlanet App Server

The Sun–Netscape Alliance, which uses the brand-name **iPlanet (www.iplanet.com)**, has begun shipping the iPlanet Web Server, Enterprise Edition 4.0, formerly known as Netscape Enterprise Server.

iPlanet Web Server EE 4.0 incorporates a multi-threaded and multi-process environment, a new integrated servlet engine, support for Java Servlet 2.1 extensions, and in-process, pluggable native JVMs. The Alliance said that based on tests conducted by **Mindcraft Inc. (www.mindcraft.com)**, Los Gatos, CA, these new features deliver performance gains of 222% in Java Servlet performance over a commercial version of the Apache Web Server, Stronghold version 2.4.2, and a 22% advantage over CGI scripts.

Wind River, Espial in Strategic Alliance

Wind River Systems (www.wrs.com) and **Espial Group Inc. (www.espial.com)** have entered into a strategic alliance to jointly push their respective technologies for Java embedded systems development.

Wind River's VxWorks RTOS, Tornado development platform, and Personal JWorks implementation of Java technology have been integrated with Espial's Kalos Architect Java technology-based embedded rapid application development tools and Internet Application Suite, which includes Espial Escape Web browser, Ebox email client, and Assistant Personal Information Manager.

Together, the two firms said their goal is to create a "one-stop software solution for building next-generation PersonalJava technology-based Internet appliances."

Wind River is based in Alameda, CA, while Espial is headquartered in Ottawa.

WHAT WERE YOU DOING THE DAY
THE COURSE OF COMPUTING
WAS ALTERED FOREVER?
Hanging
round thinking about
eing set free? We're actually
oing something about it . . . your
omputing freedom that is. At Cyrus™
ntersoft, we're working with industry
eaders on our mission to set you free. Try to
magine using a computer you've never seen
efore, in a place you've never been before, for on-
emand access to applications you've never used
efore. Presenting Speiros™ – the world's only truly
ervasive Internet computing platform. Most people
ear the phrase "anything, anytime, anywhere"
nd think freedom. We think revolution.
o learn more, visit www.cyrusintersoft.com
r call 1-877-88CYRUS.
cyrus™
INTERSOFT, INC.

Achieving Synergy

ObjectSpace's Voyager ORB 3.0

Reviewed by JuanCarlo Añez

Version Reviewed: 3.0
Current Version: 3.1.1

CUP RATING SYSTEM:
5 Outstanding **4** Very Good **3** Acceptable
2 Has Potential **1** Poor

THE FIELD OF software development evolves so rapidly and changes so often that it's nearly impossible to keep up. It's easy to feel bombarded by acronyms (CORBA, DCOM, JSP, ASP, EJB, UML) and overwhelmed by (often competing) theories, standards, and technologies.

From time to time, a coalescing idea comes along that manages to tie it all together, to leverage what exists and raise the level of abstraction a couple of notches—to find just the right synergy. The invention of structured programming languages is an example of a synergistic breakthrough; so are relational databases, graphical user interfaces (GUIs), Internetworking, and, more recently, object-oriented (OO) programming and the Java platform.

I examined a product that, if not exactly the cause of a paradigm shift, is part of this kind of synergistic breakthrough. With the Voyager ORB 3.0 from ObjectSpace, distributed computing has never been easier, cheaper, or as widely available.

Voyager is a full-blown Java ORB architecture designed to support the development of powerful distributed computing systems. The product uses the Java Virtual Machine (JVM) to load classes at runtime to create mobile objects and autonomous agents. A complete and seamless distributed computing framework, it supports remote invocation, remote pass by value, distributed events, naming services, object mobility, autonomous agents, runtime object extensions, object activation, distributed class and resource serving and loading, distributed security, distributed garbage collection, distributed timers, advanced messaging, multicasting, and replaceable networking protocols.

The Voyager framework provides all of the above with only a few abstractions, a very small API, and no need for interface definition languages (IDLs) or management of stubs or proxies. Voyager works from interfaces defined in pure Java and automatically generates and distributes whatever stubs or proxies it needs at runtime. All this functionality is provided by a single 500K JAR file of Java that users can download free (for internal use) from the ObjectSpace Web site.

The Voyager ORB comes in two versions: the Voyager ORB Core, which is free for "internal" use (it can't be distributed to third parties); and Voyager ORB Professional, the commercial version. Voyager ORB Pro includes everything in the Voyager ORB, plus a configuration framework, JNDI integration, CORBA naming service, and support for ultra-light (15K) clients. It also includes the Voyager Management Console, a Java program that enables central configuration of services offered by different clients. The just-released Voyager Pro version 3.1 (see sidebar) adds dynamic XML, connection management, persistent replicated directory, directory load balancing, and support for DCOM.

ObjectSpace sells several products that work with Voyager to provide enhanced functionality, including Voyager Transactions, Voyager Security, and the Voyager Application Server. These products provide CORBA-compatible transaction support, advanced security, and an EJB-capable application server that can be licensed separately.

OO FOR EVERYONE

Voyager truly is something of a breakthrough in distributed programming. It combines the various features of Java to produce something greater than the parts. What Voyager does has never been done before (in a commercially available product, anyway) and was almost impossible to do until the advent of Java.

The beauty of the Voyager ORB is the way it manages to hide the many complexities of distributed program-

REVIEW IN A NUTSHELL

Anyone developing distributed clients will want to check this out. It implements every imaginable distributed computing feature with only a few abstractions and a small and easy-to-use API. No other product on the market makes client programming easier.

Price:
Voyager Application Server = $1,999.00,
Voyager ORB Professional = $799.00

Vendor Info:
ObjectSpace Inc.
14850 Quorum Drive Suite 500
Dallas, TX 75240
v: 800.OBJECT.1 or 972.726.4100
f: 972.726.4200
info@objectspace.com
www.objectspace.com

It's time to say goodbye to the old "wait" problem.

PROVIDE FASTER, BETTER ACCESS TO CORPORATE DATA—FROM ABSOLUTELY ANYWHERE!

Tired of bulky solutions for client database downloads? Get your apps in shape with ThinAccess™ from thinWEB.com. When you need a fast, reliable way to deploy applets requiring database access, ThinAccess can help you:

- Download 20 to 30 times faster to lose those long delays from downloading JDBC drivers for slow network users;
- Eliminate browser restrictions, and say goodbye to solutions that work with one version but not another; and
- Access information without sacrificing security, so you can forget about problems for people connecting behind firewalls.

This hot new middleware extender installs on the server side and lets you plug in any third-party JDBC driver. The drivers remain on the server side to give you remote control. The result is ultra-thin, fast access to corporate data.

ThinAccess also provides optional connection pooling, easy connectivity to databases not on the Web server, result set processing enhancements, and pre-configuration of database queries. It's that critical missing link to Java data access over the Net!

Want to know more?
Don't let another minute go to waste—visit our Web site at www.thinweb.com, e-mail us at reach-us@thinweb.com or call 1-877-THINWEB (844-6932) ext. 227 today.

100% PURE JAVA

Java Business™ ENTERPRISE SOLUTIONS IN ACTION

thinWEB.com
all the web without the wait

ming by leveraging the strengths of the Java platform. Voyager uses introspection to discover the features of whatever class users want to make distributable (at runtime). It then generates any required wrapper classes on-the-fly by writing the bytecodes to memory and registering the just-created classes with the JVM. Voyager lets users deploy code to a single location by leveraging Java's class-loading mechanism; it supports flexible and custom security by embracing standard Java security; it supports pass-by-value and object mobility by leveraging Java serialization. All this happens under the hood, without any involvement required from the programmer. The program just makes a remote call and classes get dispatched to the right place, connections are made, and objects are serialized and reconstructed at the destination.

DYNAMIC AGGREGATION

At the core of Voyager's magic is its ability to attach secondary objects, or *facets*, to a primary object at runtime. This is what the Voyager documentation calls *Dynamic Aggregation*. Classes don't have to be modified in any way for instances to act as facets or to be extended by them.

```
IEmployee employee = new
  Employee( "joe", "234-44-2678" );
IFacets facets = Facets.of( employee );
IAccount account = (
  IAccount) facets.of(IAccount );
account.deposit( 2000 );
```

Classes used as facets do not need to be related in any way to their "primaries"; but Voyager allows different implementations of a given facet to be attached to different primaries by using an algorithm that searches for likely facet implementations using the primary's class name.

Those familiar with COM (the Microsoft Common Object Model) programming will probably recognize a similarity between COM interface querying and facets. The difference is that Voyager's Dynamic Aggregation allows for new facets to be attached to an object at runtime, even when the source code for the primary is not available. Voyager's features such as Proxies, Mobility, and Facets are implemented using Dynamic Aggregation.

REMOTE INVOCATION

In OO programming, remote invocation refers to the ability to invoke methods on remote objects (objects that exist in a different machine) as if they were local. Remote invocation is usually implemented through proxies, which provide a façade that hides the mess of the underlying plumbing. Voyager excels in this regard in that this plumbing is automatically generated at runtime, for any class, as required. This means that classes do not have to be modified in any way to be remote enabled. Most notably, they don't have to descend from a framework class nor do they have to implement a specific interface.

Method calls made to a proxy are forwarded to its related object. If the object is remote, the arguments are serialized using the standard Java serialization mechanism and reconstructed at the destination. By default, parameters are passed by value, except for actual parameters that are already proxies, or those with classes that implement

```
com.objectspace.voyager.IRemote or
java.rmi.Remote;
```

these are passed by reference. The same rules are applied for method return values. Exceptions that occur during a remote method call are caught at the remote site and rethrown locally.

Voyager also implements advanced messaging. A one-way message call returns immediately, discarding the return value if any. For "future" messages, the call returns a proxy to the result, which can be polled or read in a blocking fashion. Timeouts can be specified for future messages, and users can request that a proxy is returned in place of the actual object:

```
try {
Object args = new Object[] {"XYZ"};
Result result =
  Future.invoke(
    market, "getStock", args, true);
result.setTimeout(5000);
IStock stock = (
  IStock) result.readObject()
catch (TimeoutException e) {
  //...
```

Voyager implements distributed security on top of standard Java security. This means that security managers, permissions, signing, encryption, and custom security policies are all available to distributed applications. It also means that it is not necessary to study the intricacies of yet another security system to deploy secure distributed applications with Voyager. For additional security, Voyager allows the classes that carry out communications between hosts to be replaced with secure versions like SSL. Voyager Pro also provides a security implementation that's more aware of distribution, and supports a central configuration and remote management.

NAMING SERVICES

The usual way of getting initial references to remote objects in distributed applications is to use a naming service or registry. Voyager implements a distributed, hierarchical, and federated naming service that is tightly integrated with the rest of the framework. References to named remote objects are obtained passing the target host, port, and name to the Namespace.lookup() method:

```
IStockMarket market =
  (IStockMarket) Namespace.lookup(
    "//host:8000/NASDAQ");
```

Names can be bound to objects at creation time. The following call would bind the name NASDAQ to the newly created object and export in the default (local) server:

```
IStockMarket market =
  (IStockMarket) Factory.create(
    "StockMarket", "NASDAQ");
```

Bindings can also be done when an object is exported via the Proxy.export() method, or later, by using Name space.bind():

```
Proxy.of(market).export("NASDAQ");
```

COOL BEANS.
TheTheoryCenter™
Enterprise JavaBeans™ technology is cool stuff. What our eBusiness Smart™ Components let you do with it is even cooler. This powerful suite of components speeds up e-business application development by providing 50-75% of common ebusiness functions right out of the box—allowing you to create e-business applications in record time and focus your resources on your competitive edge. Our exclusive LaunchEJB™ program combines on-site education with the design and implementation of a usable e-business prototype that meets your individual business needs. It's the perfect way for any business to get a jump on e-business. So what are you waiting for?
JOIN US AT SUN'S JAVA BUSINESS CONFERENCE DECEMBER 6-9 AT THE JACOB JAVITS CONVENTION CENTER IN NEW YORK CITY.
Download our JumpStart™ suite of eBusiness Smart Components at www.theorycenter.com and speed your time to e-business.
For information, e-mail sales@theorycenter.com or call 1.888.843.6791
© 1999 The Theory Center. All rights reserved. The Theory Center is a trademark of The Theory Center, Inc. eBusiness Smart Components, LaunchEJB and JumpStart are trademarks of The Theory Center, Inc. Enterprise JavaBeans is a trademark of Sun Microsystems, Inc in the U.S.A. and other countries.

Oracle8*i* Lite 'Web-to-go': Extending Web Applications to Mobile Users

An Oracle Technical White Paper

Web applications have spelled good news for IT organizations: they are less costly to develop, deploy, and manage than client-server applications. Because of their increasing use of the Java language, they are faster and easier to develop and less platform dependent. Web applications are also less costly because they're entirely centrally deployed and managed — requiring no install or set-up on client devices. But to date, Web applications are only available to users with a network connection. And there has been no solution that enables mobile users to have off-line access to Web applications.

Mobile computing based on lightweight client database and replication technology is nothing new. But to this day it has relied on a client-server architecture which is challenging to developers and IT managers:

- Two different applications need to be developed: one against a lightweight single user database for mobile devices — such as Oracle8*i* Lite — and one against a central server database.
- While Web development cycles are measured in weeks, rolling out a full featured mobile application can take an order of magnitude longer, resulting in long times to market.
- Installing and maintaining the client software is expensive and can require bringing the device to IT staff.

What is really needed is a solution that makes the promise of Java and of the Internet a reality for mobile applications, a solution that lets you run the same application on a mobile device and on a server, and that provides 100% centralized deployment and maintenance of distributed software components, retaining the advantages of Internet computing.

Introducing Oracle8*i* Lite 'Web-to-go'

Oracle is proud to introduce Web-to-go, a new feature of the Oracle8*i* Lite database that brings the benefits of the Internet and Java to mobile applications. Web-to-go is a collection of components that enables users of Java and HTML based Oracle8*i* Web applications to transparently switch between online mode and mobile off-line mode. At any time, online Web application users can choose to switch to off-line mode. This triggers the automatic download and installation of the Web-to-go client, which includes Oracle8*i* Lite and a micro Web server. Oracle8*i* Lite is populated with just the right subset of data and Oracle8*i* Servlets. Users can then log off their corporate network and keep on using their Web applications — with the exact same application look and feel. Whenever users log back onto their network, powerful bi-directional replication of data and Servlets between Oracle8*i* and Oracle8*i* Lite kicks in, ensuring a transparent switch back to online mode.

Benefits of Oracle8*i* Lite 'Web-to-go'

Simplified Development

- Rather than build two separate applications, the mobile application developer only needs to create one Java Servlet application that can be used by mobile and connected users.

Key Features of Oracle8i Lite 'Web-to-go'

- Seamless online/off-line support of Java Web applications
- Automatic client setup
- 100% centralized software deployment and system management
- Enterprise scaleability, performance and reliability
- Open and simplified Web application development environment

- Developers can use any development tool that generates 100% pure Java to create the Java/HTML Servlets.

- Oracle8*i* Lite offers one unique API to the mobile application developer — with no need for mobile-specific skills, such as knowledge of replication.

Simplified Deployment and Management

IT staff never needs to handle a physical device throughout the mobile application life-cycle, cutting the cost-of-ownership of mobile applications:

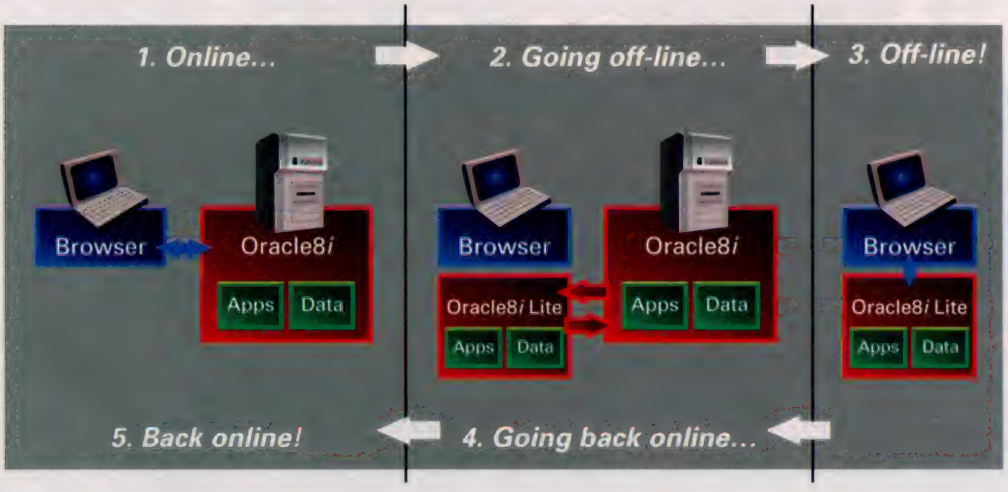
- Initial installation of client software is automatic.

- Web-to-go Servlets are stored within Oracle8*i* and Oracle8*i* Lite and replicate along with data subsets. Application updates are 100% centrally deployed: all that is needed is to update the Web-to-go Servlets in the central Oracle8*i* Application Repository once — and the new Servlets are transparently deployed to all mobile devices.

Use Web-to-go today

If you are writing Java based Web applications, writing them for Web-to-go opens them up to mobile users with little effort. For further information, or to download a demonstration, please visit the Oracle Technical Network (OTN) at http://technet.oracle.com. To activate the enclosed **FREE** 30 day trial CD of Oracle8*i* Lite, and for more information regarding Oracle8*i* Lite, visit **www.oracle.com/mobile/trylite/** or call us toll free:

USA: **1-800-ORACLE1 ext. 27064**
Other Countries: phone numbers on CD



"Oracle8i Lite elegantly solves a longstanding problem that mobile users have had with Web-based applications and continues to provide a full-featured single-user database for client/server applications. Organizations looking for a lightweight database for mobile workers or for embedded applications will find Oracle8i Lite has a lot to offer."

Timothy Dyck, *PC Week Labs Senior Analyst* — October 11, 1999

ORACLE

```
Namespace.bind("NYSE", market2);
```

Hierarchical directories can be created by using slash-separated ("/") paths for the names:

```
Namespace.bind(
  "services/exchanges/
    NYSE", market2);
```

A distributed directory can be created by binding one directory to a name in a remote host:

```
Directory exchanges =
  new Directory();
exchanges.put("NASDAQ", market1);
exchanges.put("NYSE", market2);
Namespace.bind(
  "//remotehost:8000/
    exchanges", exchanges);
```

The same API is used to access CORBA and JNDI directories just by using different URLs. For example:

```
IStockMarket market =
  (IStockMarket) Namespace.lookup(
    "rmi://host/NASDAQ");
```

A Voyager server also acts as a CORBA and JNDI naming server. Clients need only use the usual bootstrap code for naming services.

MULTICAST MECHANISM

Voyager implements an efficient multicast mechanism that supports the publish/subscribe metaphor. To participate in multicast, a program places the objects that will be receiving the messages in a special container called a *Subspace*. A Subspace can be connected to other local or remote subspaces to form a *Space*. Messages sent to a Subspace are copied to neighboring Subspaces before being delivered to objects in the local Subspace. The usual way to send a multicast message to objects in a Space is to obtain a multicast proxy of the desired type. Method calls over the proxy translate into multicast messages and eventually into calls of the same method over any object that implements the same interface:

```
ISubspace subspace = (
  ISubspace) Namespace.lookup(
    "//localhost:8000/Subspace1" );
IBroker allBrokers = (
  IBroker) subspace1.
    getMulticastProxy( "IBroker" );
allBrokers.stockChanged("XYZ");
```

Voyager's publish/subscribe allows programs to register event handlers with a space, much like it's done in Java's Abstract Windowing Toolkit (AWT). Programs subscribe to specific topics by adding instances of the Subscriber class to a space. Topics are strings or names separated by periods, like "stock.XYZ.up." A Subscriber acts on messages that match the topic pattern passed to its constructor and forwards the received event to its associated PublishedEventListener.

OBJECT MOBILITY

Voyager allows remotely-enabled objects to be moved from host to host. Mobility allows moving objects that exchange large numbers of messages closer to each other, reducing network traffic. Mobility can also be used for server balancing and for clustering. Moving an object in Voyager is very simple:

```
IStockMarket market =
  (IStockMarket)
    Namespace.lookup("NASDAQ");
Mobility.of(market).moveTo(
  "//another_server:7000");
```

Objects with classes that implement Voyager's IMobile interface will receive pre- and post-notifications of departures and arrivals when they are moved.

Voyager provides a special type of mobility in the form of agents. Agents can roam around the network by themselves carrying the necessary information to load their related classes and other resources. When an agent is set to be autonomous, it won't be garbage collected even after there are no more references to it.

```
IBroker broker =
  Factory.create("Broker");
IAgent agent = Agent.of(broker);
agent.setAutonomous(true);
agent.setResourceLoader(
  new URLResourceLoader(
    "http://thishost:8000"));
agent.moveTo("//markethost",
  "roamAndBuy");
```

Existing proxies to objects that are moved are updated the next time they are used.

RESOURCE SERVING

Making sure that the Java class files for a distributed application are in sync on all the deployment hosts can be a nightmare. Voyager solves this problem by allowing any HTTP (Web) server as a source for class files and other resources:

```
VoyagerClassLoader.addURLResource(
  "http://serverforclasses/path");
VoyagerClassLoader.
  addURLResource(
    "http://yetanotherserver");
```

Voyager itself can act as an HTTP server, so there's no need to deploy any other software for serving class files. Voyager also allows for classes to be loaded from signed JAR files or from custom class loaders that implement the IResourceLoader interface.

With resource serving, all the classes that implement a distributed application's functionality can be deployed to a single host or a pool of them. Resource serving also allows for mobile agents to roam around a network, arriving at hosts on which class files required for running the agent would normally not be available.

ACTIVATION

Object activation means the ability to store objects to permanent storage and resurrect them afterward on request, even when the underlying program has been restarted. Voyager's activation architecture is independent of the persistence mechanism used. To enable activation for a given class of objects, a program provides an activator and registers it with Voyager's activation manager.

```
ILibrary library = new Library();
LibraryActivator activator =
  new LibraryActivator(library);
Activation.register( activator );
Namespace.bind( "Library", library );
```

An activator is an object whose class

VisualCafé™ 4

Expert Edition

It's new. It's here. And it's only $299.95* for a competitive upgrade.

New quick configuration for Java™ 2, JDK™ 1.1 and JDK 1.3 Beta
New team source code control management
Even faster Java 2 execution
New native Java 2 applications

*Suggested retail price is $799. The $299.95 competitive upgrade price is available to licensed owners of Inprise™ J-Builder, IBM® VisualAge,® Microsoft® J++, Sybase® PowerJ,® Oracle® JDeveloper,™ Asymetrix® Supercede,® and Sun Microsystems® Java Workshop.™ Previous ownership of one of the competitive products listed above is required. Limited time offer only in USA.

CompUSA® • Computer Discount Warehouse™ • Fry's Electronics® • J&R Computer World® • MICRO CENTER®
MicroWAREHOUSE® • PC Connection® • Programmer's Paradise® • PROVANTAGE®

For details, visit us now at www.visualcafe.com/vc4/offer/eecompupgrade.

VisualCafé is a trademark of Symantec Corporation. Java, JDK, and all Java based marks are trademarks of Sun Microsystems, Inc. Other brands and products are trademarks of their respective holder/s. Copyright © 1999 Symantec Corporation. All Rights Reserved.

The #1-selling Java™ development tool. **www.visualcafe.com**

Building a Simple Distributed App

The following example is designed to give you a taste of the Voyager APIs and to demonstrate how simple it can be to program distributed applications. Running Voyager is as simple as setting the CLASSPATH and executing the distributed application. Voyager also includes a standalone server that can be executed from the command line.

These are the steps I went through to build a simple distributed application that exercises object mobility:

1. Define a Java interface that describes the objects we'll be moving around:

```
public interface ITalker {
  void say(String message);
}
```

2. Provide a straightforward implementation of the interface:

```
public class Talker
implements ITalker,
java.io.Serializable {
    public void say(String message) {
        System.out.println(message);
    }
}
```

The class must implement java.io.Serializable to be able to move around the 'Net.

3. Compile the two Java files and start the Voyager ORB:

```
voyager 8000 -r
```

The number (800), tells Voyager which port receives requests. The "-r" parameter tells the ORB to act as an HTTP server for classes in its CLASSPATH.

4. Copy the ITalker.class to a different machine and write, compile, and run the following Java program:

```
import com.objectspace.voyager.*;
import com.objectspace.voyager.
  loader.*;
import com.objectspace.voyager.
  mobility.*;

public class Houdini {
    // change "momo" to the name
of the actual remote machine
    static final String peer =
"//momo:8000";

    public static void main(String[]
args) throws Exception {
        // allow required classes to be
loaded from peer

VoyagerClassLoader.addURLResource(
  "http:" +peer);
        // start the local Voyager ORB
        Voyager.startup();
        try {
          // create an instance of our
ITalker implementation
          ITalker talker = (ITalker)
            Factory.create("Talker");
          // do the magic
          talker.say("1,2...");

Mobility.of(talker).moveTo(peer);
          talker.say("3! Applause
            please!");

System.out.println("Applause
  please!");
        } finally {
          Voyager.shutdown();
        }
    }
}
```

This small program started a client ORB, created an object from a class file provided by a different machine, moved the object to another machine, and remotely invoked a method on the (now) remote object. I used a text editor, a Java compiler, the Java Runtime Environment, and Voyager. There was no IDL, and all the required proxies, stubs, and other plumbing were provided by Voyager at runtime by leveraging the networking, serialization, and introspection capabilities of Java.

implements the IActivator interface. It is responsible for providing a string that acts as a key and uniquely identifies activated objects and for reconstructing the original objects when requested. Voyager provides support for taking snapshots of activated objects and of their facets. After an object has been enabled for activation, the process of activation is transparent to clients that hold references to activated objects.

Given the simplicity of Voyager programming, its CORBA features are especially appealing. I tested the product's compatibility with the Visigenic ORB (VisiBroker) bundled with Inprise's JBuilder Enterprise 3. In the first test, a VisiBroker client tried to interact with the server program from Voyager's simple "Bank" example. The second test used Voyager as a client against VisiBroker's own Bank example.

To perform the first test, I copied the files from Voyager's Bank example to a different directory. Then IDL files were produced from the Java interfaces and exception classes using Voyager's CGEN utility. The IDL files were then used to generate new Java interfaces and CORBA stubs and skeletons using VisiBroker's IDL2JAVA tool. Then a client was written using the VisiBroker libraries. Both programs were compiled, the server was started, and the client was run. This experiment didn't work. The VisiBroker client halted with an exception no matter what I tried.

The second test applied the same process to VisiBroker's Bank example. This time the experiment ran on the first try.

CONCLUSION

The only real complaint I have about Voyager ORB is the lack of documentation available about ORB semantics. I've heard complaints from other users that the ORB has scalability problems due to the way it assigns threads to requests, but my experience with the product revealed no such shortcoming. In any event, that kind of problem is easy enough to foresee and simple enough to avoid with the right kind of documentation.

I strongly recommend this product to anyone developing distributed clients. No other product I'm familiar with makes client programming easier. Developers working on distributed applications, especially those involving applets and/or servlets, also would love this product. And anyone who wants to experiment and learn about distributed computing could hardly go wrong downloading this product and taking it out for a spin. Voyager ORB Core is free, after all. It's easy to install, simple to learn, and even simpler to use. ■

Juancarlo Añez is director of IT at Modelistica in Caracas, Venezuela. He can be contacted at juancarlo@modelistica.com.

Java™ 2 Platform Enterprise Edition

FOR THE REAL WORLD

Java™ 2 Platform Enterprise Edition (J2EE) is the single most important standard to emerge since Web applications became the world's new way of doing business. Understanding this standard and how to implement it is critical to the success of your Web applications. Now you can learn what you need to know about the J2EE platform – and how to make it all work for you.

SEMINAR SCHEDULE

Amsterdam	Dec. 1
Atlanta	Dec. 1
Boston	Nov. 3
Brussels	Nov. 30
Chicago	Nov. 16
Dallas	Dec. 8
Denver	Nov. 10
Houston	Dec. 7
London	Nov. 16
Morristown, NJ	Nov. 4
New York City	Nov. 3
Paris	Dec. 2
Washington, DC	Nov. 30

Attend a FREE half-day educational seminar – *Java™ 2 Platform Enterprise Edition for the Real World,* brought to you by SilverStream Software and Sun Microsystems. Through technical demonstrations and instruction from industry experts, you'll gain a better understanding of the impact of the J2EE platform, the details of the specification and how you can use it to quickly deliver robust Web solutions.

WHY ATTEND?

- Learn about using the J2EE platform to implement real-world applications
- Discover how and when to use J2EE technologies such as Enterprise JavaBeans™ components (EJB), Servlets, JavaServer Pages™ components (JSP), Java™ Transaction Service (JTS) and Java™ Foundation Classes (including the GUI toolkit, JFC/Swing)
- Receive a Beta of the newest version of the SilverStream Application Server
- Get valuable materials to reference as you implement your applications
- Learn how to access additional resources such as training and services to help you get started

WHO SHOULD ATTEND?

Whether you're a developer or architect who's already knowledgeable about the J2EE platform, or you're just starting to investigate it, this FREE half-day educational seminar is right for you.

REGISTER TODAY FOR A FREE HALF-DAY EDUCATIONAL SEMINAR

www.silverstream.com

Or phone 1.888.823.9700
(toll-free in the U.S.)
or 1.508.788.3260
(outside the U.S.)

GOLD SPONSOR

Java Business Show
Visit our booth for J2EE info, and FREE software!*
December 7-9
Jacob Javits Center, NY
www.javabusinessconference.com

*SilverStream *Inside Look* CD includes a 45-day evaluation copy of the newest version of the SilverStream Application Server, plus a product tour!

LEARN HOW TO DELIVER REAL-WORLD APPLICATIONS WITH THE J2EE PLATFORM.

Copyright ©1999 SilverStream Software, Inc. SilverStream is a registered trademark and SilverStream Designer is a trademark of SilverStream Software, Inc. Sun, Sun Microsystems, the Sun logo, Java, Enterprise JavaBeans, and JavaServer Pages are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Sun microsystems | SilverStream®

Java Server Pages and Servlet Design

How to build maintainable systems more quickly

JOHN AKERLEY,
ALEX KOUTSOUMBOS,
MURTUZA HASHIM,
AND ANGELO MAFFIONE

Alan Witschonke/Images.com

WE EXAMINE SOME of the design issues uncovered during a project building a demonstration Internet banking system at the San Jose International Technical Support Organization (ITSO) at IBM.

During the design of this system and in examining similar Web applications, we found several applications of design patterns including Command, Facade, and Adaptor as well as uses of the Model View Controller (MVC) and domain firewall paradigms.

One of the biggest choices in designing Web applications, as in many others, is how to break the application into components or subsystems that are easier to create, understand, and maintain. In many Smalltalk- and Java-based systems the MVC paradigm is used.

MVC was originally defined as the classes used in Smalltalk to build user interfaces (UIs) where view objects listened for changes to a model object and updated the UI view in response to these changes. The controller managed the interaction between the user and the interface.

There are now many flavors of MVC implementations. What they have in common is the separation of the model or business logic from the view logic in an application. In many cases the controller will manage the interaction between the model and the view. This is the model we used for our Web application.

Most Web applications are very similar to the transaction-based applications running on mainframe systems. You fill in fields on a terminal screen and then submit it for processing. Another screen is then sent to you depending on the values you entered in the previous screen. The difference between this Web application model and the traditional MVC is that the view does not listen for changes to the model; the view is updated by the controller with data from the model based on user input.

The model in our implementation was implemented using

need it? get it. tools | code | tips | how-to's | tutorials | Security | Java™ | Linux® | Web | XML

ibm.com/developerWorks

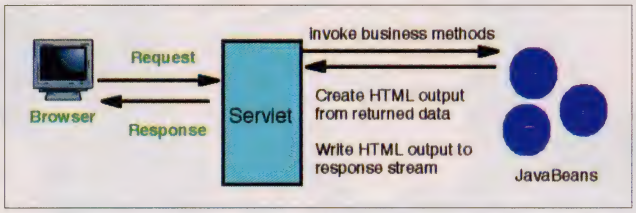


Figure 1. ***Servlet as Controller and View.***

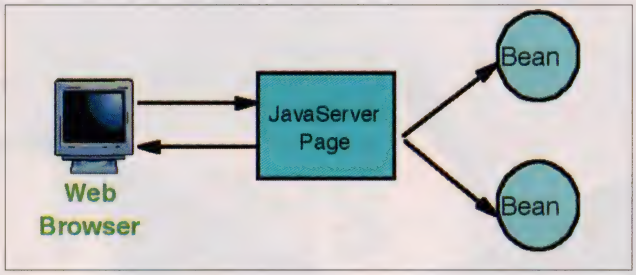


Figure 2. ***Access Model 1.***

JavaBeans, which provided the bank's business logic. The back end of the application could be implemented using many different approaches, including Enterprise JavaBeans (EJB) or JDBC. The following design approaches are largely independent of the approach used to implement the business logic.

In using this flavor of MVC to build our application, we found four basic design questions:

1. Where do you put the controller?
2. How do you access your business logic?
3. How do you return the results to the client?
4. What should you put in a Java Server Page (JSP)?

Now let's answer these questions. We used the JSP 0.91 specification as implemented in the IBM WebSphere Application Server 2.0 but the approaches and issues also translate to the 0.92 and 1.0 implementations of JSPs.

Where Do You Put the Controller?

As shown in Figure 1, in existing servlet-based Web applications the servlet is used as both the controller and the view (HTML is coded in Java or read from a file), but this has disadvantages. If the servlet is used to generate dynamic content, then any changes made to the format of the output also requires that the servlet be recompiled. This makes the application more difficult to maintain, particularly if there are frequent changes to the format of the output. It's also quite difficult to create sophisticated Web pages using HTML output from servlets. Using JSPs, the view is a JSP and the controller can either be a servlet or a JSP.

In the JSP 0.92 specification these two approaches are defined as Access Model 1 and Access Model 2. In Model 1 the controller is the JSP. In Model 2 the controller is a servlet. Although these model definitions have been removed from the JSP 1.0 specification and replaced with high level architecture descriptions, they are still relevant to designing Web applications.

Model 1—The Controller Is a JSP

In Model 1 a browser directly accesses a JSP (Figure 2).

Using Model 1, a JSP for a transfer from a bank account might look like Listing 1 (assume the existence of a bank object).

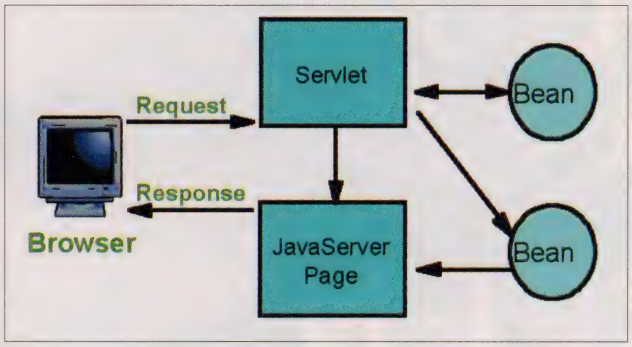


Figure 3. ***Access Model 2.***

The logic in this example must determine if the transfer is permissible and then perform the transfer and display the confirmation or display an error. This results in conditional, or controller, code in the page as well as the view code. This may make maintenance of the application more difficult. In addition, the tools currently available for Java development do not provide strong support for JSP/HTML and vice-versa. However, using JSPs as the controller may result in less code written than the Model 2 approach and might be a good choice for small or prototype projects or in projects where the view is fairly simple.

Model 2—The Controller Is a Servlet

In Model 2 a browser accesses a JSP indirectly through a servlet which interacts with the business logic, possibly through JavaBeans (Figure 3). After receiving the client request, the servlet performs any necessary computation and inserts the appropriate JavaBeans into the JSP, which is then invoked.

The JSP extracts whatever information it requires from the JavaBeans and merges them with the HTML page. The browser then interprets and renders the HTML as usual.

The same example of the funds transfer using Model 1 (Listing 1) could be separated into two JSPs and one servlet. The first JSP is the Funds Transferred page and might look like Listing 2.

The second JSP would simply report that there were insufficient funds for the transfer. The servlet would perform the transfer and direct the control to the appropriate page (note that the `callPage` method has been deprecated and replaced by the forward method on the `RequestDispatcher` interface). See the Servlet example code in Listing 3.

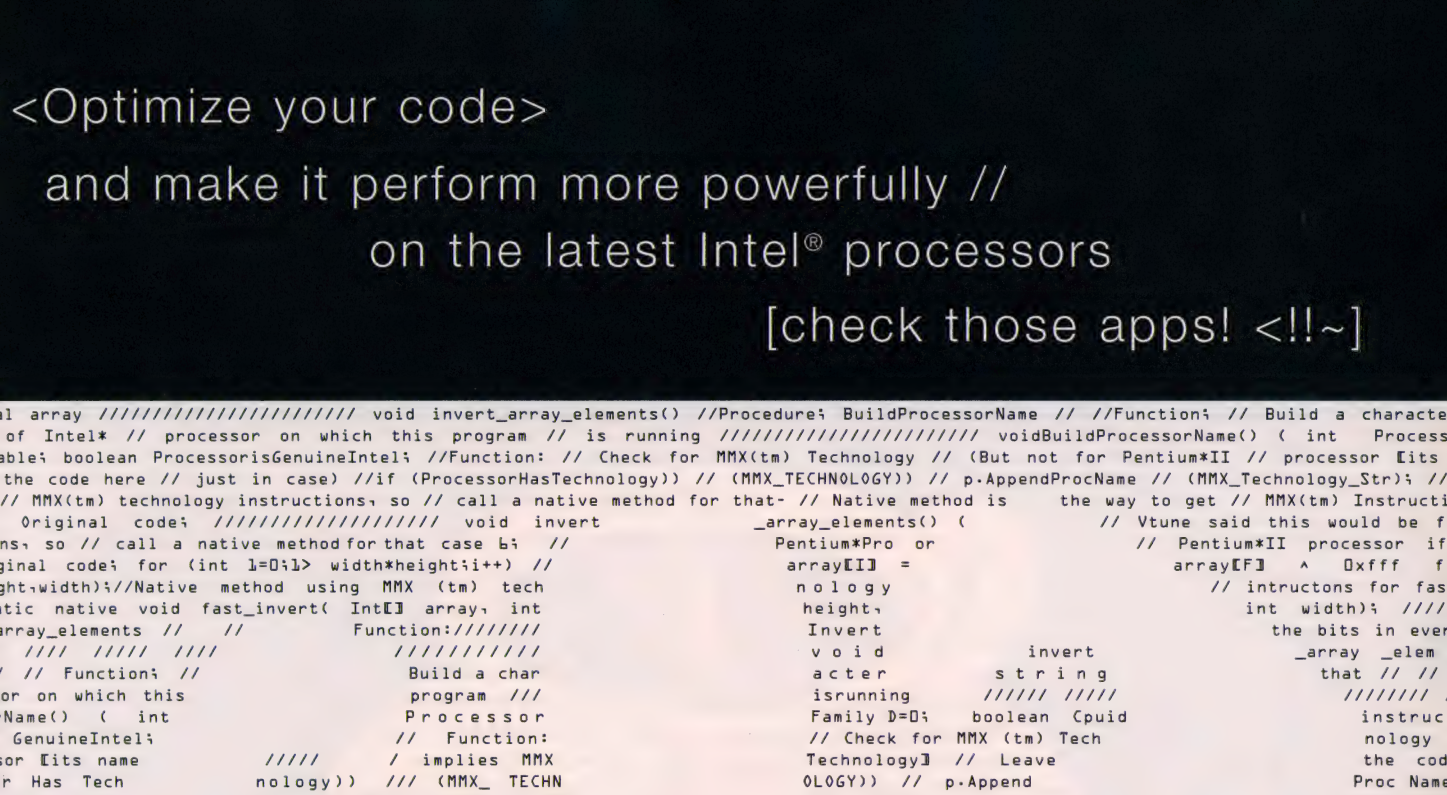
While using servlets as the controller adds another layer to the application beyond JSPs, we think it is a good idea because:

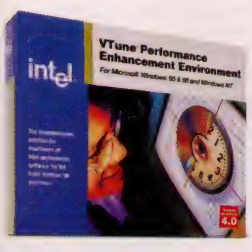
- View and controller code are logically separated.
- You can use the best tools for Java and for HTML/JSP without trying to combine them.
- Your Java developers and HTML developers are not working on the same code.

How Do You Access Business Logic?

The purpose of most Web applications is to invoke business functions and extend these business applications to users on the Web. In many cases this function resides on other systems and in legacy applications. How your application accesses this business logic is another issue in your design. The application can directly invoke these functions, but this

<Optimize your code>
and make it perform more powerfully // on the latest Intel® processors
[check those apps! <!!~]


l array ////////////////////////// void invert_array_elements() //Procedure; BuildProcessorName // //Function; // Build a characte
of Intel* // processor on which this program // is running ////////////////////// voidBuildProcessorName() (int Process
able; boolean ProcessorisGenuineIntel; //Function: // Check for MMX(tm) Technology // (But not for Pentium*II // processor [its
the code here // just in case) //if (ProcessorHasTechnology)) // (MMX_TECHNOLOGY)) // p.AppendProcName // (MMX_Technology_Str); //
// MMX(tm) technology instructions, so // call a native method for that- // Native method is the way to get // MMX(tm) Instructi
Original code; ////////////////////// void invert _array_elements() (// Vtune said this would be f
ns, so // call a native method for that case 6; // Pentium*Pro or // Pentium*II processor if
ginal code; for (int 1=0;1> width*height;i++) // array[I] = array[F] ^ 0xfff f
ght,width);//Native method using MMX (tm) tech nology // intructons for fas
tic native void fast_invert(Int[] array, int height, int width); ////
array_elements // // Function:////////// Invert the bits in ever
//// ///// //// //////////// invert _array _elem
// Function; // Build a char string that // //
or on which this program /// ////// ///// ////////
Name() (int Processor Cpuid instruc
GenuineIntel; // Function: // Check for MMX (tm) Tech nology
sor [its name ///// / implies MMX Technology] // Leave the cod
r Has Tech nology)) /// (MMX_ TECHN 0LOGY)) // p.Append Proc Name
this loop would be faster with // MMX (tm) techn ology inst
that- // Native method is the way to get // MMX(tm) Instruction
Original code; ////////// ///// ////// void invert _arr
uld be faster with // MMX(tm) //// /// technology i
d for // // /// that case 6; if// Instructio
ssor (int 1=0;1 >w
for Oxfff fff; //
^ ,height ,width); /
ay intructons for fas
// native void fast_inv
tic /////////////// /////
th); // Function: // Invert
s // array //// //// ////////
mensional //Procedure; Build ProcessorNa
ements() represents the type of Inte
ring that // // ///// //// //// ///// void Bu
is running ////// Cpuidinstruction Avail
FamilyD=0; boolean MMX(tm) Technology // (But no
tel; //Function: // Check for code here // just
/ implies MMX Technology] // Leave the ProcName // (MMX_Technology_Str
gy)) // (MMX_TECHNOLOGY)) // p.Append instructions, so // call a nat
ster with // MMX(tm) technology into Java(TM) // applications
the way to get // MMX(tm) Instructions said this would be faster with
void invert_array_elements() (// Vtune Pentium*Pro or // Pentium*I
all a native method for that case 6; // 1=0;1>width*height;i++) // array
ster array // Original code; for (int //Native method using MMX (tm) tech
sion fast_invert(array,height,width); fast_invert(Int[] array, int he
ment // inversion public static native void Function: // Invert the bits in ever
/// //Procedure; Invert_array_elements // // //Procedure; BuildProcessorName // //
//////////////////// void invert_array_elements() on which this program // is runni
// // represents the type of Intel* // processor CpuidinstructionAvailable; boolean Pro
ocessorName() (int ProcessorFamilyD=0; boolean // processor [its name // implies MMX
for MMX(tm) Technology // (But not for Pentium*II (MMX_TECHNOLOGY)) // p.Append Pr
just in case) //if (ProcessorHasTechnology)) //



//Here's the easy way to increase your code's power, especially if you're pressed for time. It's the VTune™ Performance Enhancement Environment 4.0. All the tools you need to take advantage of the power of the latest Intel processors, including the Pentium® III processor. The VTune Performance Analyzer quickly identifies performance bottlenecks and delivers priceless tuning advice. The high-performance Intel C/C ++ Compiler features support for the Pentium III processor's Streaming SIMD Extensions, and plugs into Microsoft Visual Studio.* And if that wasn't enough to whip your code into perfect shape, there's the Intel Performance Library Suite and the Intel Performance Training Center. Not surprisingly, this simple way to add extra muscle to your apps comes from the people who know the processor best. So optimize your code and make it as powerful as you imagined. < Close dialogue now >

Download a free, 14-day evaluation @ http://developer.intel.com/vtune

intel®

©1999 Intel Corporation. All rights reserved. Intel and Pentium are registered trademarks, and VTune is a trademark of Intel Corporation.
*All other brands and trade names are the property of their respective owners.

Table 1. Domain Firewall and Comand Pattern comparison.

	DOMAIN FIREWALL	COMMAND PATTERN
PROS	Object-oriented interface	One point of entry
	Understandable domain	Easy to implement logging and undo actions
	Supports different client types	Matches well with transactional systems
		Maps well to Use Case models
CONS	May be complex to implement	Not an intuitive domain interface
	Many points of entry	Loss of object-oriented model

approach makes changes to the code difficult. Separating your application code from the business logic will create a more maintainable and easily tested application. Two possible methods to use are:

- Command Pattern (Figure 4)—Used to package and execute commands and is very effective when you want the commands to be executed uniformly but don't know where or how they necessarily will be executed.

In designing a Web application that uses the Command Pattern the use cases map roughly to the commands. For each use case, you create a command that is executed in a Command handler.

- Domain Firewall (Figure 5)—Is related to the Facade design pattern. It is a class or interface (or set of classes or interfaces) representing the business model. It provides all the functionality of the business objects without exposing the application to the implementation of these objects.

Table 1 compares the two approaches.

How Do You Return the Results to the Client?

A third design issue is the form in which results (the dynamic content created by interacting with the model) are returned to the client. JSPs can interact directly with your business logic and extract information for the user. However, this will link your JSP code closely with your business logic and may surface code that the view should not be accessing.

LISTING 1.

Model 1 example code.

```
<!— Standard HTML headers/titles would go here >
<H1>Withdrawal</H1>
// Java code to retrieve account numbers and amount
<%
  String amount = request.getParameter("amount");
  String fromAccountId =
                    request.getParameter("fromAccountId");
  String toAccountId = request.getParameter("toAccountId");
  try{
      bank.transfer( fromAccountId, toAccountId, amount);
  }
  catch( InsufficientFundsException e){
      out.println("<H2>Error withdrawing funds</H2>");
      out.println("<P>Return to the previous page an try again");
  }
  out.println("<H1>Funds Transferred</H1>");
  out.println("<p>Transferred " + amount + " from " +
  fromAccountId);
%>
```

A solution to this problem is to employ the Adaptor pattern (see Figure 6). Adaptor beans encapsulate the results of actions on the business logic and are built to be handled either through Java code in JSPs or using JSP bean tag syntax.

In our project we created adaptor, or view, beans to be used in JSPs. These beans are created by the servlet and placed in the session or the request. They simply encapsulate data based on the results of accessing the business logic.

As with using servlets as controllers, using adaptor beans will add another layer of code to your application but will make the code cleaner and easier to maintain. This separation will also make your application easier to test, because you can unit-test each subsystem using dummy data in the view beans. An additional benefit of adaptor beans is that they can be used for any transforms required between the model and the view, to support internationalization, for example. The following is an example of using an adaptor bean to wrap the results of a function. A transaction record is placed in the JSP to be displayed after the user has deposited money into his/her bank account:

```
TransactionRecord record = account.deposit( 100.);
TransactionRecordView view = new TransactionRecordView( record);
request.putValue( view);
```

An additional reason to use adaptor beans is that, using the current JSP implementations, a bean in a JSP must be defined as a class; it cannot be defined as an instance of an object implementing an interface, and any methods that throw checked exceptions must be caught in the JSP.

What Goes Into a JSP?

Now you know that we think the servlet should act as the controller and you should use adaptor beans to pass results to the view. You still need to decide what exactly goes into

LISTING 2.

Model 2 example—funds transferred JSP.

```
<H1>Funds Transferred</H1>
<BEAN NAME="fromAccountId" TYPE="String" INTROSPECT="no"
CREATE="no"
  SCOPE="request"></BEAN>
<BEAN NAME="amount" TYPE="String" INTROSPECT="no"
CREATE="no"
  SCOPE="request"></BEAN>
<p>Transferred <insert bean=amount></insert>
from <insert bean=fromAccountId></insert>
<!— HTML code showing transfer details>
```

Banking on Java?

Today's Retail Banks have to bring new products to market in ever more challenging timeframes. Your IT Staff face equal challenges - to deliver flexible, easily maintained systems to meet fast-changing business requirements. BankFrame from **EON Technologies** offers all the ingredients you need to build scalable, mission-critical Enterprise Java Banking solutions.

Watch out for EJB for Finance Seminars across America in November.
Contact us today on (781) 229 5848

Visit us at BAI's Retail Delivery '99, December 7-10, Miami Beach, Florida
Booth Number: 388. You can also see us at the Sun Microsystems' Booth, Number 913.

BankFrame™

The world leader in Java Banking Solutions

Offices: USA, Ireland, UK, South Africa, India www.bankframe.com info@bankframe.com

Table 2. JSP beans tag specification changes.

WebSphere 0.91 Implementation	JSP 0.92 Specification	JSP 1.0 Specification
BEAN	USEBEAN	jsp:useBean
INSERT	DISPLAY	jsp:getProperty
REPEAT	LOOP	deferred

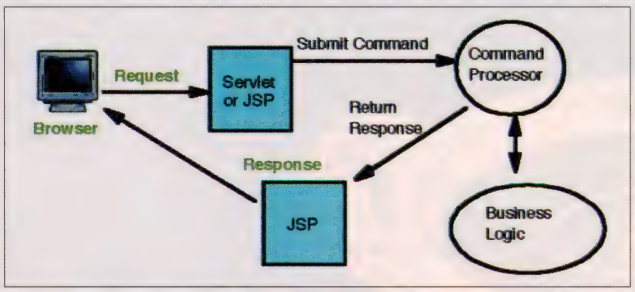


Figure 4. ***Command Pattern used with JSPs.***

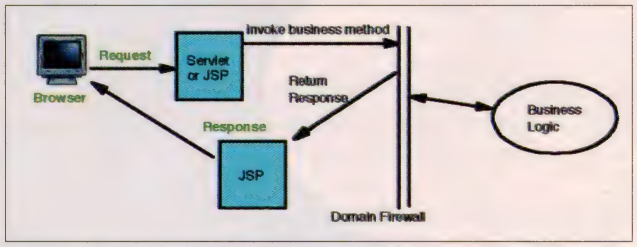


Figure 5. ***Domain Firewall used with JSPs.***

a JSP. If you have chosen to use Model 2 a JSP can contain Java code or bean tags to represent dynamic content. If you have chosen Model 1 you will probably need to have Java code within the JSP.

You can embed Java code in the JSP using scriptlets. This approach is appropriate when the programming logic is relatively minor. Having any more than a trivial amount of programming logic inside the HTML document makes it more difficult to separate roles in a development team as the HTML and Java code are tightly coupled. This is not to say that scriptlets don't have their uses, but we believe their use should be minimized and they should only be used where it does not make sense for this code to reside inside components.

In the second approach, the programming logic resides in components and bean tags are used to request information from these components, which is then inserted into the HTML document. This approach has the advantage in that it facilitates a cleaner separation of roles in a development team and the programming logic is contained inside components, which would be the responsibility of a Java programmer.

LISTING 3.

Model 2 example—Servlet.

```
String amount = request.getParameter("amount");
String fromAccountId = request.getParameter("fromAccountId");
String toAccountId = request.getParameter("toAccountId");
try{
  bank.transfer( fromAccountId, toAccountId, amount);
}
catch( InsufficientFundsException e){
  request.callPage( "error-page.jsp");
}
request.callPage( "funds-transferred.jsp");
```

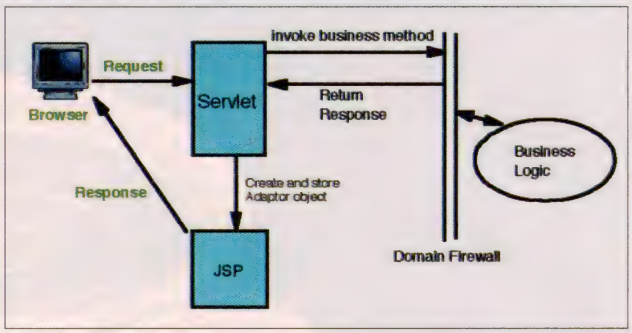


Figure 6. ***Adaptor pattern with JSPs.***

The bean tag syntax has changed considerably in the latest revisions of the JSP specification. Table 2 shows the corresponding tags in the different levels of JSP.

The loop construct is now deferred until a standard tag extension is agreed to. Several application servers currently support different tag extensions, and the intention is to define a standard way to extend the specification to support tags.

Conclusion

JSPs are a new technology (even by Web standards) and the tools and techniques are evolving. Naming the patterns and paradigms possible in Web applications will help you communicate and discuss design ideas and recognize these patterns and paradigms in other Web applications.

As you work with the current implementations, thinking through these design issues should help you build maintainable systems more quickly.

Acknowledgments

Thanks to the members of the Software Group Technology Center at IBM Austin, who created the Freeside bank application and introduced us to the use of the Command Pattern in Web applications.

This article is the result of a residency at the International Technical Support Organization (ITSO) at IBM. ITSO residencies produce books and workshops to help IBM employees and customers use IBM products. Go to **www.redbooks.ibm.com** for more information about residencies and workshops. ■

John Akerley (j_akerley@yahoo.com) is an engineer at eHealth Insurance.com and was the project leader for this residency. The residents were Murtuza Hashim (hashimm@us.ibm.com), an e-Business IT Specialist in IBM Global Services US; Alexander Koutsoumbos (alexk@au1.ibm.com), a technical consultant for IBM Australia; Angelo Maffione (a_maffione@it.ibm.com), an I/T specialist at the Java Technology Center in Bari, Italy.

COMPONENTS BY DESIGN™ is an auction-based model for outsourcing software component development.

Get to know a planet-wide community of qualified developers at Flashline.com. Post your component specification and pre-screen developers through discussions about your specific project.

With Components by Design, you can expand your knowledgebase without expanding your programming staff.

Need a component?
Visit flashline.com. Use the coupon code JR1299 to waive the $100 posting fee. Valid through December 31, 1999.

FLASHLINE.COM
THE SOFTWARE COMPONENT MARKETPLACE

WWW.FLASHLINE.COM • 216.861.4000 • INFO@FLASHLINE.COM

Kamalova/Images.com

Advanced JavaBeans Development

Enhancing a hierarchical graphical tree component

ELDON METZ

JAVABEANS IS A powerful component model. You can build a simple Bean with little effort; with further effort you can add features that significantly aid the Bean assembler during design. For example, by simply following the naming conventions to create getter/setter pairs for private data members, you can build a basic JavaBeans component that can be integrated and assembled within any JavaBeans-capable IDE. Taking that a step further, you can enhance the assembly experience dramatically as we will do here.

We guide you through the implementation of the more advanced features of the JavaBeans specification by enhancing a hierarchical graphical tree (see Figure 1) component represented by class TreeControl. As you recall, a *tree control* is a display structure that manages nodes and subtrees. Nodes contain both images and text; subtrees can expand and collapse.

In the JavaBeans specification, any Java class with the proper getters and setters can be recognized as a JavaBean by any Beans builder-aware IDE. To make our tree control Bean stand out in a crowd, however, we add the delegation-style event model, a custom property editor, and a WYSIWYG customizer. Finally, we discuss how to package our JavaBean, so that the customer will know how to separate the runtime components from the design-time components when deploying the assembled application.

Extending the Event Model for Useful JavaBeans Communication

To begin, let's look at the delegation-event model. One of the most important design issues for our Bean is making sure that our component exposes events correctly so that other components can properly catch and process the events. To determine which events should be exposed, we

Racing to develop a more compelling Web application?

Now you can easily deploy Java™ thin-client user interfaces instead of drowning in HTML.

Classic Blend™

Classic Blend™ consists of an ultra-thin Java client that displays your user interface and a communication mechanism for connecting the client to any middle-tier Java application server. Classic Blend™ is the first tool to support a true ultra-thin client architecture by its unique functional partitioning, placing the display on the client and the presentation and application logic on the application server.

- Classic Blend simplifies the development of n-tiered systems by automating the client-application server connection...so you don't have to. You simply develop a two-tiered system and Classic Blend gives you the last tier for free!
- Classic Blend works with any Java bean-based IDE and complements leading Java application servers such as BEA Systems WebLogic®.
- Users will love the desktop-like experience of a more interactive and dynamic Java user interface.
- Classic Blend user interfaces update faster because less data is moved over the network. No more "blank screens" as HTML refreshes.
- Classic Blend-powered systems scale better than HTML-based systems because the server does less work by eliminating the generation and transmission of large HTML files.

Classic Blend™ by AppliedReasoning

Making Ultra-Thin Java User Interfaces a Reality Today.

Win the race to the Web with Classic Blend™.

Download FREE evaluation software!
www.appliedreasoning.com

AppliedReasoning™

www.appliedreasoning.com

800-260-2772

913-319-0900

Copyright 1999 Applied Reasoning Systems Corporation. All rights reserved. Applied Reasoning, Classic Blend, the Classic Blend logo and the Applied Reasoning logo are all trademarks of Applied Reasoning Systems Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Applied Reasoning Systems Corporation is independent of Sun Microsystems, Inc. All other trademarks are the property of their respective owners.

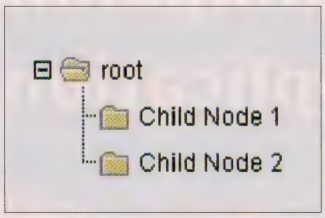


Figure 1. ***A graphical tree.***

identify the actions within the tree control that may require notification to an external component. For example, events are generated when a node is selected or deselected or when a node that contains children is collapsed or expanded.

The delegation-event model consists of a *source* of events which can have any number of *listeners*; listeners are notified when an action on the source causes the emission of a particular type of event. For our tree control, we create a TreeNodeEvent and a TreeNodeListener. The TreeNodeEvent, which derives from the java.awt.AWTEvent class, identifies the type of event, the node from which the event originated, and the input event that caused the event to happen (keyboard or mouse). For convenience, we also create a *TreeNodeAdapter* to provide default implementations for each of the events. With the adapter, the user need only implement the methods for the events they choose.

Creating a TreeNodeEvent. Figure 2 shows the derivation of TreeNodeEvent, the tree node semantic event; the code is in Listing 1.

Creating a TreeNodeListener. The TreeNodeListener interface extends the java.util.EventListener tagging interface that all listeners must extend. The derivation of TreeNodeListener is shown in Figure 3; the code for the listener interface is in Listing 2.

Adding the registration methods. Now that we have established the foundation for the event model, we add the TreeNodeEvent registration methods to the TreeControl class, as shown in Listing 3. These methods are add TreeNodeListener() and removeTreeNodeListener(). Internally, we identify the places where events should be fired so that all registered listeners can receive them.

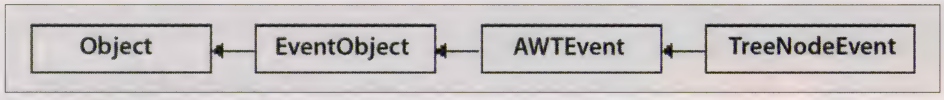


Figure 2. ***The derivation of*** **TreeNodeEvent.**

Firing an event. In the internals of the tree control, the

LISTING 1.

Semantic event code.

```
package com.roguewave.widgets.tree.v3_0;

import java.awt.AWTEvent;
import java.awt.Event;
import java.awt.event.InputEvent;

/**
 * Class TreeNodeEvent is the tree node semantic event.
 * @see com.roguewave.widgets.v3_0.tree.TreeNodeListener
 *
 */
public class TreeNodeEvent extends AWTEvent {

   private TreeNode node;
   private InputEvent event;

   /**
    * Indicates a tree node has been expanded.
    */
   public final static int TREE_NODE_EXPANDED= 0;

   /**
    * Indicates a tree node has been contracted.
    */
   public final static int TREE_NODE_CONTRACTED= 1;

   /**
    * Indicates a tree node has been selected.
    */
   public final static int TREE_NODE_SELECTED  = 2;

   /**
    * Indicates a tree node has been deselected.
    */
   public final static int TREE_NODE_DESELECTED= 3;

   /**
    * Indicates a tree node has been clicked.
    */
   public final static int TREE_NODE_CLICKED   = 4;


   /**
    * Constructs a TreeNodeEvent object with the specified source
    * object.
    * @param source The object where the event originated
    * @param id The type of event
    * @param node The node
    */
   public TreeNodeEvent(Object source, int id,
                        InputEvent event, TreeNode node) {
     super(source, id);
     this.event = event;
     this.node = node;
   }

   /**
    * Gets the TreeNode from this event.
    *
    * @return The node sent with this event.
    */
   public TreeNode getTreeNode() {
     return node;
   }

   /**
    * Gets the InputEvent from this event (if any).
    *
    *@return The input event that caused this event,
    *or null if none.
    */
   public InputEvent getInputEvent() {
     return event;
   }
}
```

ADVERTISEMENT

NAMING VARIABLES

by Adam Kolawa

This month we will examine our first dangerous construct in Java.

Code should be written as clearly as possible so that developers can easily determine its function and debug its algorithms. Therefore, you should choose variable names that connect variables to their real-life functions.

However, developers often use abbreviations as shortcuts when they name variables. A class that adds and subtracts money from a bank account might appear as follows:

```
public class ba {
   public static String curr = "dollars";

   public void dep (int i) {
      bal += i;
   }
   public void wit (int i) {
      bal -= i;
   }
   public String get () {
      return Integer.toString (bal) + " " + curr;
   }

   private int bal;
}
```

At a glance, nobody can tell what the above code does. Developers won't automatically assume that ba represents a bank account, and while they are supposed to be checking the function of the code they will be forced to make a translation for each variable. When the human brain is occupied by more than one task at a time, it is more likely to make errors.

Standard naming conventions help connect code to its real-life function and free the mind to focus on one problem on a time, which is the key to successful error prevention. The code should be re-written as follows:

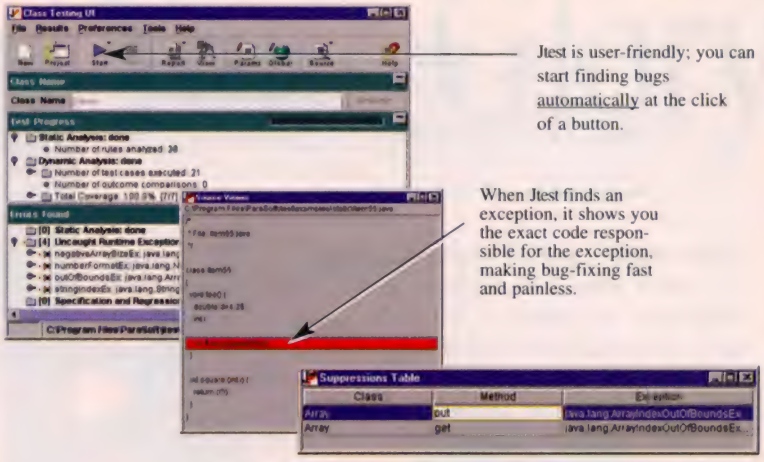
```
public class BankAccount  {
   public static String CURRENCY = "dollars";

   public void deposit (int amount) {
      _balance += amount;
   }
   public void withdraw (int amount) {
      _balance -= amount;
   }
   public String getBalance () {
      return Integer.toString (_balance) + " " + CURRENCY;
   }

   private int _balance;
}
```

As you can see, naming conventions reduce confusion among developers and increase the maintainability of your code. The few extra seconds you spend using standard naming conventions today can save hours of extra work for you and your development group later in the development cycle.

Adam Kolawa is President and CEO of ParaSoft. He can be reached at ukola@parasoft.com



Jtest is user-friendly; you can start finding bugs automatically at the click of a button.

When Jtest finds an exception, it shows you the exact code responsible for the exception, making bug-fixing fast and painless.

With Jtest, it's easy to focus on the exceptions that are most important to you. You can customize reports to suppress exceptions by type, name, and enclosing method.

Would you believe that Jtest® can take your brand-new class and test it on its own, automatically?

"Jtest® represents a dramatic step forward in Java™ testing. As developers move toward serious application development, it is absolutely critical for them to incorporate tools such as Jtest in their development cycle." -- Will Iverson, Symantec

Jtest® is the world's first automatic class testing tool for Java™. Testing on its own with one click from the developer, Jtest automatically passes unexpected inputs to each class and validates that the inputs will not cause the program to crash. Using patented Test Generation System Technology (#5,784,553 & 5,761,408), Jtest tests all methods of a class. When Jtest finds an input that generates an uncaught runtime exception, it reports to the developer the arguments that cause the problem, the stack trace for the exception found, and the calling sequence that shows the exception happening. The testing process with Jtest prevents errors that could cause problems when classes are integrated into modules.

For version 3.0, ParaSoft has added three impressive new testing capabilities to Jtest: black-box testing, regression testing, and static analysis.

Black-box testing, which ensures that each class behaves according to specification, is now a quick and easy process with Jtest. During black-box testing, Jtest automatically provides a set of inputs based on sophisticated analysis and executes the class with the inputs. The user can also provide a set of inputs that will be tested along with the automatically generated inputs. Jtest will then report the outcomes, which the user can verify with one click of the mouse.

Jtest automates regression testing, a process similar to black-box testing. During regression testing, Jtest helps make sure that developers do not re-introduce errors to their code. Jtest remembers outcomes from previous runs and compares outcomes every time the class is tested, reporting an error if an outcome changes. Developers can use Jtest to make sure they are not letting their code regress into previous errors when they make changes.

In addition, you can now enforce Java coding standards automatically with the new static analysis feature of Jtest. Jtest parses the Java source of a class and applies to it a set of rules. Any violations to these rules are reported as errors. Enforcing coding standards at every stage of the development process will lessen the amount of debugging developers must perform later in the development cycle.

If you would like the satisfaction of using Jtest, visit www.parasoft.com/errors.htm or call ParaSoft toll-free at (888)305-0041 for more information.

ParaSoft®

www.parasoft.com/errors.htm

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. ParaSoft is a registered trademark and Jtest is a registered trademark of the ParaSoft Corporation.

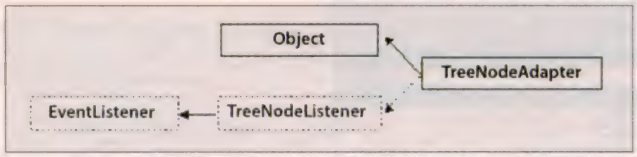


Figure 3. ***The derivation of* TreeNodeListener.**

fireTreeNode() method shown in Listing 3 is called when an event should be emitted to all registered listeners. The code in Listing 4 (Listings 4–10 are available at *Java Report Online*—www.javareport.com) is taken from the TreeNode class. When a node is clicked, the tree control's selection manager invokes the setSelected() method with a boolean. Once this method is called, logic is executed to determine whether the node is currently being selected or deselected. The appropriate event is then fired to all registered listeners.

Wiring our Bean. Now that we have added the delegation-event model, we can wire our Bean with other Beans so that when an event happens, the listener Beans can take some action. The screenshot in Figure 4 illustrates the wiring mechanism we will use, which is provided by the BDK. The BDK is the reference testbed for examining JavaBeans components and how they interact with others.

In the BDK, reflection is used to determine whether the

LISTING 2.

The listener interface.

```
package com.roguewave.widgets.tree.v3_0;

/**
 * This is the listener interface for
 * receiving tree node change notification.
 */
public interface TreeNodeListener extends java.util.EventListener {

   /**
    * Is invoked when a TreeNode has been expanded.
    */
   public void nodeExpanded(TreeNodeEvent event);

   /**
    * Is invoked when a TreeNode has been contracted.
    */
   public void nodeContracted(TreeNodeEvent event);

   /**
    * Is invoked when a TreeNode has been selected.
    */
   public void nodeSelected(TreeNodeEvent event);

   /**
    * Is invoked when a TreeNode has been deselected.
    */
   public void nodeDeselected(TreeNodeEvent event);

   /**
    * Is invoked when a TreeNode has been clicked.
    */
   public void nodeClicked(TreeNodeEvent event);
}
```

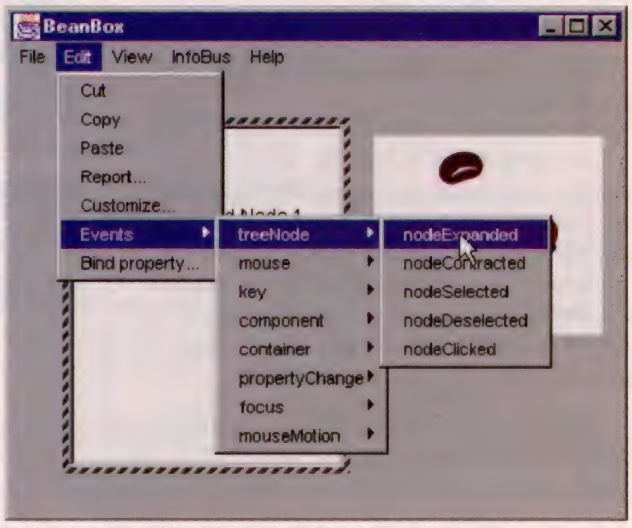


Figure 4. ***Choosing the tree node event to wire.***

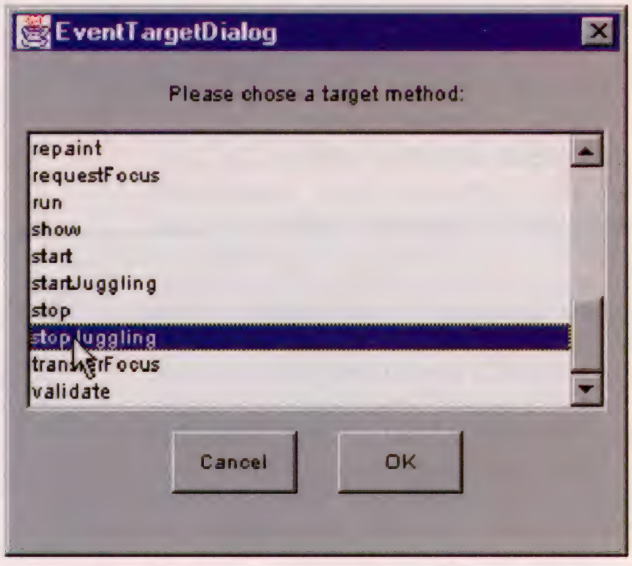


Figure 5. ***Choosing the method to invoke on the target Bean.***

TreeControl JavaBean has a listener, and to identify each supported event to be added to the allowed choices in the related menu structure. Commercial Bean builders vary in their level of support for wiring components together. Some help by showing you which events to connect but let you complete the work yourself by taking you directly to the source code to write the event handling code. Other Bean builders let you perform the entire task visually by drawing lines between components, then modifying the properties of each particular connection line. There are no major interoperability issues with the event model. The wiring interface does vary dramatically from builder to builder.

Returning now to wiring our Bean in the BDK, we first select the Bean that is the source of events, then select the event that should be handled. This is shown in Figure 4.

At this point, we must select a Bean to wire to. We choose the juggler, which brings up a list of methods with no arguments or methods that take the event object, TreeNodeEvent, as a single parameter. These are shown in Figure 5.

Our Guts. Your Glory.

"GemStone/J will become the centerpiece of our advanced money transfer product. GemStone is clearly the application server leader."

— Richard Leurig, VP and CIO, MoneyGram and Travelers Express

NEW
White Paper on Internet Commerce by Industry Analyst John Rymer!

You're open for business on the World Wide Web. Sophisticated, custom-built Internet commerce applications are driving your business into greater growth and profitability. Life's looking good at the top. ▼ But are these applications robust and secure? They are, if you build them on the GemStone/J application server. It has the most comprehensive security solution available and the guts to really power your applications. World-class stuff built on the Java 2 security standards as well as integrated EJB and CORBA-based plumbing. Just ask our customers. ▼ Check out what a leading industry analyst, John Rymer of Upstream Consulting, thinks in his white paper **"Charting the Course of Internet Commerce Architectures."** Call 800-243-9369 or get it at **www.gemstone.com/guts**. Your glory may depend on it.

GEMSTONE®
So you can serve the world

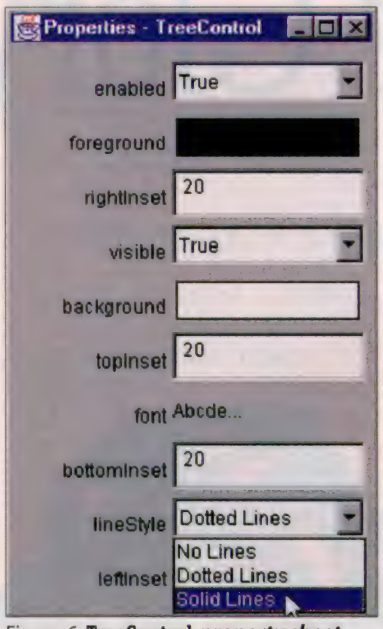


Figure 6. **TreeControl *property sheet illustrating* lineStyle *property editor.***

Some Beans builders, IBM's VisualAge in particular, provide quite a bit of flexibility when wiring a connection like this. In the example explained above, when a node in the tree control is expanded, the juggler is notified and stops juggling. In real life assembly, wiring scenarios are much different, as the tree node event is usually queried with custom code to determine the action to be performed.

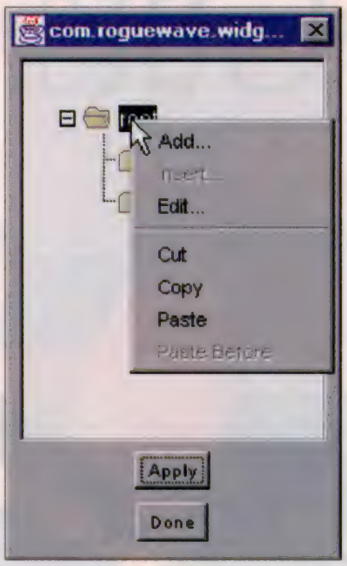


Figure 7. **TreeControl *customizer in action.***

Creating a Custom Property Editor

We continue enhancing our graphical tree component by adding a custom property editor. To understand the value of this feature, let's consider how it can be used.

TreeControl can create a tree control structure with a solid line, a dotted line, or no lines between nodes. Bean builders usually present this decision to the designer in the property sheet, where it can be implemented in a number of ways: as a text property, where the user types in the correct string, for example; or as two boolean properties, something like hasLines() and isSolid(). If the designer chooses the latter, however, the isSolid() boolean might not be valid, depending on the hasLines() boolean. The ideal solution is to use constants that represent the line style, and to have a single property that lets the user select from a fixed set of choices. Because it's not user-friendly to make Bean assemblers enter constants to pick the right one (0, 1, 2, etc.), we provide a custom property editor. The Bean builder lets the user interact with the property editor in the property sheet, as shown in Figure 6.

Creating a lineStyle property. The first step in creating our custom property editor is to create a lineStyle property. To set this up, we first create the constants

LISTING 3.

Registration methods

```
public class TreeControl extends ScrollPane implements  Cloneable {

 ...

 /**
  * Adds the specified tree node listener to receive tree node events
  * from this TreeControl.
  *
  * @param listener The tree node listener
  */
 public synchronized void
 addTreeNodeListener(TreeNodeListener listener) {
   listeners.addElement(listener);
 }


 /**
  * Removes the specified tree node listener so it no longer
  * receives tree node events from this TreeControl.
  *
  * @param listener The tree node listener
  */
 public synchronized void
 removeTreeNodeListener(TreeNodeListener listener) {
   listeners.removeElement(listener);
 }


 /**
  * Fires the specified tree node event to all listeners.
  *
  * @param event The tree node event that should be fired.
  */
 protected void fireTreeNodeEvent(TreeNodeEvent event) {
   Vector l;
   synchronized (this) { l = (Vector)listeners.clone(); }

   for (int i = 0; i < l.size(); i++) {
     switch (event.getID()) {
     case TreeNodeEvent.TREE_NODE_EXPANDED:
       ((TreeNodeListener)l.elementAt(i)).nodeExpanded(event);
       break;
     case TreeNodeEvent.TREE_NODE_CONTRACTED:
       ((TreeNodeListener)l.elementAt(i)).nodeContracted(event);
       break;
     case TreeNodeEvent.TREE_NODE_SELECTED:
       ((TreeNodeListener)l.elementAt(i)).nodeSelected(event);
       break;
     case TreeNodeEvent.TREE_NODE_DESELECTED:
       ((TreeNodeListener)l.elementAt(i)).nodeDeselected(event);
       break;
     case TreeNodeEvent.TREE_NODE_CLICKED:
       ((TreeNodeListener)l.elementAt(i)).nodeClicked(event);
       break;
     }
   }
 }

 ...
}
```

HERE'S WHAT YOU WANT.

SYBASE®
INFORMATION ANYWHERE.™

Our customers wanted all of the above, too. And they got it with Sybase Enterprise Application Server.™ Check out the success of Blue Cross/Blue Shield, First American, and others at www.sybase.com/easerversuccess. Or call Sybase at 1-800-8-SYBASE. And make everybody happy. Especially yourself.

©1999 Sybase, Inc. All rights reserved. All trademarks are the property of their respective owners.

and add the setter/getter for the lineStyle property that sets and returns an integer value. The code is shown in Listing 5.

Implementing the Custom Property Editor. The next step is to implement our custom property editor by extending the PropertyEditorSupport class, which is provided in the java.beans package. We need implement only three methods: getTags(), setAsText(), and getAsText(). Implementing these methods creates a property editor consisting of a choice of strings that the user can select. When the user selects a choice, the value of the choice is compared with some internal constants, which are then mapped to the correct integer value to update the lineStyle property. Likewise, when a Bean is selected and the property sheet displays the current state of the TreeControl, the lineStyle property is read from the Bean as an integer, then mapped to the correct string representation. The user immediately sees a human readable translation of the constant integer selection. The code for our custom property editor is given in Listing 6.

Assigning the Custom Property Editor to the lineStyle property. Once we have created a lineStyle property and implemented our custom property editor, specifying the property editor to use in the BeanInfo class is straightforward. After creating the PropertyDescriptor in the getPropertyDescriptors() method, the PropertyEditor class is set to the one that you have created, as shown in Listing 7.

Custom Property Editors: benefits and cautions. As you may deduce from our example, many useful custom property editors can be implemented. Outside the current paper, we created a property editor that employs a slider to allow users to select from a valid range of values. We also wrote a custom property editor that allows users to choose images from the file system or from any valid URL location by clicking on thumbnails displayed right in the property sheet. The major benefit of choosing the right custom property editor is that the user will find that your Bean simplifies the design and assembly process. Good custom property editors can also make the process of manipulating and assembling Beans a more enjoyable task.

Be careful, however, when you are providing custom property editors for objects other than primitives. Almost all the Bean builders invoke getJavaInitalizationString() to properly persist a selected change in source code. Because most Bean builders use code generation rather than serialization to persist property changes, it is important that the Bean builder can get the proper string to initialize the custom object in the state that you have selected.

Creating a WYSIWYG Customizer

The JavaBeans specification allows the Bean designer to provide a customizer, the final enhancement to our graphical tree control. A customizer is useful when you want to make changes to a Bean that cannot be performed by a simple property change. A customizer can also provide a wizard-like interface for entering multiple properties, if they must be entered in some required order. Depending on the Bean, the requirements will vary, and many Beans may not require a customizer at all.

For our tree control example, we provide a WYSIWYG customizer that allows the Bean assembler to create and remove nodes, as well as to move them around and change their content, both images and text. To make these changes easy, in WYSIWYG fashion, be sure that the objects used for the nodes are cloneable.

Creating the customizer. To provide a customizer, we must first create a class that implements the *java.beans.Customizer* interface. The customizer interface is not complex; it consists of only three methods: one that sets the object to be customized, and two that register property change listeners so that the IDE is aware of property changes taking place. Our tree control customizer only uses the property change mechanism to notify the builder that something has changed, so that a repaint or any other necessary action can occur.

We will add content to the tree, which is not a property in the JavaBean's interface, so we will not fire property change events that the builder could use to replicate the changes in source code form. Because the IDE cannot capture these changes, we must inform it that we are making changes that can only be replicated by serializing and deserializing the modified object before making any other customizations. The hidden-state changes are specified in the BeanInfo class; they were added to a later version of the specification after interoperability issues arose. An example of how to specify that the state will be hidden in the BeanInfo class is shown in Listing 8.

We must still use the property change mechanism to inform the IDE that a change has occurred. Implementing the methods addPropertyChangeListener() and removePropertyChangeListener() is most easily done by reusing the convenient PropertyChangeSupport class provided in the java.beans package. This class can be a private member in our customizer to which addPropertyChangeListener() and removePropertyChangeListener() can directly delegate. See the code in Listing 9.

The focus for developing our customizer is the setObject() method. In the constructor for the customizer, we set a few booleans for managing the state of the customizer to their initial value; we also set the layout manager. Then, when a tree control JavaBean instance is selected to be customized, the setObject() method is called and the instance is passed as a parameter. At that point, we create a pop-up menu and a tree control named clone. The clone object sets its properties to that of the instance passed in, and clones the tree nodes so that it looks exactly like the tree control being assembled. Users can then modify the tree as they like. The rest of the code, shown in Listing 9, deals with interacting with the pop-up menu selections.

Applying customizer changes. As the Bean designer, we have several options for when to apply our changes.

Celebrating our 11th Successful Year!

C++ WORLD

The Users Conference and Exhibition

Mark Your Calendar

December 1-3, 1999
Biscayne Bay Marriott • Miami, FL

The pages of C++ REPORT come to life for 3 info-packed days.
4 Tracks, over 30 sessions –
presented by the leading writers/thinkers in the language.

Created Specially for Advanced Users Building Enterprise-Wide Distributed Systems.

Attend this 3-day Bootcamp with the C++ Gurus.

Designed for power users of large-scale systems. Join us in Miami Beach and learn the cutting edge techniques and proven solutions from the top instructors in the business. We will carefully guide you through all the technical "in's & out's" and traps & pitfalls of the language – ensuring success in your current C++ projects.

You will discuss innovative programming techniques, design approaches and program tips, and you'll network with a world-class faculty. Share experiences with others like yourself- those who are building the largest scale C++ systems in the world.

This intensive educational bootcamp is geared for serious C++ programmers and developers–those with 3+ years experience, regardless of platform or industry.

Customize Your Program

The technical program is designed to give each attendee the flexibility to mix and match their classes, providing customized training and the continued education needed for advancement. Choose from over 30 carefully selected sessions in all, including: *Advanced Principles of OOD in UML, Tips for Optimizing "Plain" & "Multi-threaded" C++ Code* and *C++ Gotchas.*

The Pages of C++ Report Come to Life

C++ World is the longest running and the only cross-platform C++ focused event available. Year after year, our commitment to the educational requirements of our delegates remains solid. Class sessions are created with your needs in mind, and present the most up-to-date, technically accurate information.

Attend the Conference and Enjoy Miami Over the Weekend!

Conveniently located (just 12 minutes from the airport) and standing watch over a 220-slip marina on beautiful Biscayne Bay, the Marriott features jet skis, deep-sea fishing, and scuba diving. You relax in the pool and whirlpool overlooking the bay. Take a swing at the golf course, or nearby tennis courts. The Miami Seaquarium and Viscaya Gardens are neighboring attractions.

What You'll Learn:

Build on your existing C++ foundation through a wide spectrum of educational sessions and workshops. 38 sessions to choose from.

- judicious use of C++ exception handling
- how to move effectively to Java
- efficient use of the template technique with patterns
- understand how patterns relate to one another
- learn compound patterns, structurally similar patterns, structurally inverted patterns & patterns
- to navigate away from C++ gotchas in coding, design patterns & idioms, and project management
- understand the advanced features of STL
- how to use RTTI properly
- how to really use function pointers and function like objects in C++ idioms and designs
- the "in's & out's" of COM+
- how to apply UML to an OO design
- secrets for making your code exception safe
- how to optimize your C++ code

Night School - Enhance Your Program Selection

Why stop when the sun goes down? Enhance your learning and take away by attending evening classes on Thursday December 2. Choose from among 3 intensive sessions – 1 1/2 hours in duration with a light dinner. Simply add them to your course selection and extend your learning into the PM.

Who Should Attend?

Designed specifically for you, the advanced C++ developers and programmers. We looked at what your needs honed in on the most important content and targeted the best speakers. The result: an intensive, learning packed program created just for you. The following audience will get the most from this conference:

- Programmers, developers, engineers
- Software architects & system analysts
- Managers of Intranet/Web Development
- Project Managers/Leaders
- Consultants & Leaders

9 Compelling Reasons to Attend

1. You get the guidelines and advice that significantly improve your chances of success with C++ projects. No other show offers the depth & breadth of C++ related topics.
2. Learn directly from the creators and implementors of C++: Stephen Dewhurst, Herb Sutter and Stan Lippman.
3. A fresh technical program- while the "winners" return, over 80% of our classes are new. Get the latest thinking in COM+, patterns, CORBA 3.0, STL, patterns, eXtreme programming, UML.1.1 & more!
4. Objective and vendor neutral. We go beyond the product hype and give you a reality check. You get the practical specifics needed to build high quality C++ systems.
5. Condense years of man hours of research and expenses into 3 fast paced days. Get pertinent "how-to" advice you can put to work immediately.
6. A great location for work & play. Enjoy the benefits of a balmy Miami Beach and the luxurious Miami Beach Marriott Resort in December. Extend your trip to enjoy the weekend and specially reduced hotel rates.
7. Learn from the best in the business. Never before have such an elite faculty been assembled to teach C++. Members of our faculty have authored the best selling books, helped build the largest systems, and are the most sought-after consultants.
8. From a very dependable source. Organized by SIGS Conferences, C++ World has an 11 year tradition of providing the highest rated technical programs in C++ found anywhere. Benefit from our long-standing commitment to the language.
9. The well respected pages of C++ Report come to life as many of our writers and editors such as Rob Martin, Andy Koenig, Barbara Moo, Herb Sutter, Stan Lippman, and more, come to Miami Beach to share their perspectives and ideas with you.

Attention Project/Training Managers!

Take Advantage of Our Team Discount and Save $200...

With 4 tracks and over 30 sessions, how can you be certain to get the most out of **C++ World**? Cover your bases – send several members of your team. You'll get the maximum benefit when 3 or more of your team members attend. See the registration form or our web site for all the ways to save: **www.cwconference.com**.

For more information:

phone: (212) 242.7515 • fax: (212) 242.7578 • e-mail: conferences@sigs.com Or visit our web site at: www.cwconference.com for more information

Sponsored by: SIGS CONFERENCES, 101 communications

Media sponsors: Java Report, C++ Report, Windows Developer's Journal, Dr. Dobb's Journal, JOOP, C/C++ Users Journal, MSJ, Application Development Trends

We could update the changes as we go, but for this particular customizer, we thought it more useful to apply changes to the target instance at times we choose. The bottom portion of the customizer panel contains an Apply button, shown in Figure 7. When this button is pressed, the nodes in the tree control in the customizer are cloned and set in the target tree control. A property support event is fired so that the IDE is aware that the target has changed and can repaint the IDE's designer window. The customizer window is still open so the user can continue making changes, applying them, and so on.

Testing the customizer. If you are testing the customizer in the BeanBox, you will see a Done button located below the Apply button that we provide. We cannot capture the user's input to this Done button, as the interface may vary between Bean builders. The biggest drawback is that, if the user presses the Done button accidentally before pressing the Apply button, all his changes will be lost. Hopefully a better solution, perhaps an interface that requires the customizer implementor to handle the standard set of responses, will be proposed in a later version of the JavaBeans specification. For now, if you are creating customizers and not saving changes as you go, be sure to create a button or other control to warn the user to manually save changes.

Packaging Our JavaBean

At this point, we have completed the enhancements to our graphical tree control component and it's time to package up our Bean. There are three useful JAR files that can be create to help the Bean assembler properly use our JavaBean. At a minimum, we should provide two runtime JARs and a single design-time JAR.

Contents of the design-time JAR. The design-time JAR file should contain all the necessary design-time classes, such as the bean info class, the customizer, and the property editors; the palette icons; and the runtime files, such as the tree control class and its supporting classes. For many of our JavaBean components, we also include HTML documentation on how to use the JavaBean in the design-time JAR file. The JavaBeans specification allows the Bean designer to include a top-level HTML file with the same name as the JavaBean. The HTML file can then reference any documentation that you prefer, whether local to the JAR file or in a URL.

In our own experience, we have found that, although we routinely place the user's guide and HTML documentation inside our design-time JavaBeans components, we have yet to see a JavaBeans IDE make use of it! Adding the documentation to the design-time JAR file makes it quite a bit larger than the runtime JAR file, as the HTML files usually contain graphics that take up a lot of space. The design-time JAR file is imported into a Beans builder, and the Beans builder usually allows you to add it to a desired location on the palette.

Contents of the runtime JARs. The runtime JAR file should contain only runtime class files, the bare minimum, yet essential classes necessary to run the JavaBean. When the Bean assembler finishes designing an applet or application, the runtime JAR file can be included in the Applet tag for download, or in the CLASSPATH for an application. To work in a CLASSPATH, a runtime JAR must be uncompressed. If you are creating a runtime JAR for downloading to an applet, you should definitely compress it to minimize bandwidth consumption.

To easily partition our classes, we show the tree control GNU makefile in Listing 10 that we used to cleanly split the right components into their respective JAR files. This makefile will create three JAR files: one compressed design-time JAR file and two runtime JAR files, one compressed and one uncompressed. Note that before this makefile was executed, the documentation-related files were copied to the directory where the make was executed. The files were removed immediately afterward by a shell script that executed a series of these makefiles for a collection of JavaBeans components.

Conclusion

Developing JavaBeans components can be as complex as you, the Bean developer, choose to make it. The nice thing about the JavaBeans component model is that simple features are simple, and yet more powerful features are available with a little more effort.

Depending on the behavior of your JavaBean, be sure to expose the right events so that other Beans can effectively communicate with your Bean. If you have properties that must be listed as choices to map internally to the Bean as constants, or if you have first-class objects that don't have property editors, consider supplying a custom property editor to let users modify properties through the property sheet of the Beans builder tool. Implementing a customizer is also an option, if you want to allow Bean assemblers to modify internals of your Bean that are not exposed as properties or to modify a set of properties presented in a logical order. This kind of a sequence, known as a wizard, is a common feature provided by applications today.

Finally, test your JavaBean in several major JavaBeans builders on the market today to ensure that all the features work as you intend them. Deliver your Bean in two forms, runtime and design-time, to aid the consumer during assembly and deployment of the application. ■

Eldon Metz is a Java and C++ developer with Rogue Wave Software in Corvallis, OR. He can be contacted at metz@roguewave.com.

URL

A free download of the BDK
www.javasoft.com/beans

Quantity reprints of this article can be purchased by phone: 717.399.1900 ext. 139 or by email: sales@rmsreprints.com.

HAVE YOU EVER TRIED TO PUT EXCEL ON THE WEB?

Sure, users love Excel. It's too bad that a desktop-bound application and the Internet just don't fit together very easily.

But with Formula One, the analytical spreadsheet reporting tool for distributed computing environments, users can leverage Excel-like power and functionality in database reporting and analysis applications. Formula One offers significant advantages over Excel when used on the Web:

- **Formula One is built in the Java programming language and is ideal for widespread distribution on the Web.** Excel and Office Web Components are designed for use behind a firewall and require Microsoft Office to be installed on all desktops.
- **Formula One is 100% Pure Java and only requires the proper virtual machine to function.** Excel requires Windows to operate.
- **Formula One provides a JavaBean and applet with an API of more than 400 properties, methods, and events.** Excel is not an API-driven application and can not be used as a component in a Java application.
- **Formula One has a lightweight footprint.** System requirements for Excel 2000 include 146 MB of hard disk space. A Formula One JAR file is approximately 1 MB.
- **Formula One writes files optimized for the Web.** Formula One's file compression technology is built with distributed computing in mind, enabling it to save files up to 90% smaller than Excel's in some cases. Excel's overhead caused by its dependence on Office and Windows produces files not suited for Web-based computing.
- **Formula One's architecture and JDBC methods enable it to be used with a wide variety of database and application servers.** Excel can't.

NEXT TIME, USE BUSINESS INTELLIGENCE DESIGNED FOR THE WEB.

FORMULA ONE 7.0

Analytical Spreadsheet Reporting Tool For Distributed Computing Environments

- ***Cross-platform***
- ***Light footprint***
- ***Extensive API***
- ***JDBC methods***
- ***Excel compatible***
- ***Charting***

INFORMATION:
800-884-8665

DOWNLOAD FREE TRIAL:
WWW.TIDESTONE.COM

A PRODUCT OF Tidestone TECHNOLOGIES

Copyright © 1999 Tidestone Technologies, Inc. All rights reserved. Tidestone, the Tidestone logo, and Formula One, are trademarks of Tidestone Technologies, Inc.
Java, 100% Pure Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All other trademarks are property of their respective owners. Specifications subject to change without notice.

Object-Oriented Widgets

Matt Foster/Images.com

Solving user interface problems by extending an AWT and a Swing Widget to make them act more object-oriented

ROBB SHECTER

A WHILE AGO I had a problem with a program I was writing. I was working on a dialog for a simple database application with contact information. The dialog was to display a list of names from the database, and when a user clicked on one of the names, detailed information about the person would be shown. I was using the AWT List class, and the more I thought about the problem, the harder it got. When an item got selected, the List class would report the first and last name of the person. In that case, my program would not receive the unique ID of the person—and to retrieve it, I'd have to somehow parse out the user's name and do a query. In order to make the parsing and query easier, I might have to display the names in the list like "LastName, FirstName," instead of the way I had originally wanted the user to see them: "FirstName LastName." And, what would happen if two users had the same name? The quick solution that first came to mind was to maintain a parallel Vector of Person objects or Person ID numbers. The program would add a Person object to this Vector every time it put his/her name in the list. Then, when a name was selected, I could fish out the real information from the Vector. But this would get messy fast. I would have to make sure that every piece of code that made changes to the List would also make changes to this parallel Vector. I also didn't like this solution because I'd seen the results of similar hacks done in very un-object-oriented (OO) C programming.

The true problem was that although the Java class libraries are OO, they only partially support OO programming. The AWT List widget only works with strings, so it's difficult to use it to view and select objects. The Swing JTable class can also be difficult to use. The information you want to display has to be somehow put into a TableModel and then plugged into a JTable. Programming in a truly OO style may mean implementing several different TableModels that

MOBILIZE YOUR APPLICATIONS

JUMPING BEANS

lets you quickly create applications which dynamically jump from computer to computer during execution, in a secure, robust fashion.

Fully exploit the capabilities of Java with Jumping Beans, an intuitive and powerful mobility framework allowing never-before-possible solutions in:

- System management,
- Database access,
- Business process implementation,
- Workflow,
- Data gathering and inventory,
- and much more...

1999 OOP '99 MÜNCHEN BEST OF SHOW Best Framework

1998 SIGS Conference JAVA Development BEST OF SHOW Best Java Frameworks

FULL-FEATURED EVALUATION COPY AT:
www.JumpingBeans.com
OR CALL 408-738-4616

"Jumping Beans" and the Bouncing Bean Logo are trademarks of Ad Astra Engineering, Inc.
Java, 100% Pure Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries

extract the information you want to show. Put another way, the problem is similar to the "impedance mismatch" that occurs when relational databases are used with OO systems; only part of a program works easily and natively with objects, forcing them to be "flattened out" into scalar values, and then later "unflattened" back into true objects.

After realizing what my real problem was, I considered how I'd like to use widgets and objects. Here's what I came up with: It would be cool if I could take a bunch of objects and throw them to a widget, which would then display them in the best way it could. The widget would let the user select one or more of the objects. If the program asked the widget for the thing that got selected, it would get a reference to a real object, not just a description of it. An advanced widget would even let the user manipulate or edit the objects in some way.

I solved several user interface (UI) problems by extending an AWT and a Swing widget to make them act more OO. The work I describe here can easily be applied to other UI classes.

Extending the AWT List

The AWT List class has the following methods (among others) for adding and retrieving strings:

- addItem(String)
- getSelectedItem()

The AWT List class would be much more usable if it had a method like addObject(Object, String), which would add the given object to its list and represent it with the given String. It could also have a method addObject(Object) as a convenience that would call Object.toString() to obtain the String representation. Finally, it would have getSelectedObject(), parallel to the original getSelectedItem().

Implementing this was fairly easy: I just applied "standard" OO refactoring to my original hack. If I had kept on using the "parallel Vector" idea, I would have ended up with lots of applications, all using instances of List and all maintaining their own parallel Vectors. My solution was to have a special List class that maintains its own parallel Vector internally, eliminating redundant coding. The Vector would be private, and manipulated when a public method is called. Subclassing the List class was the best way to implement this. My new OOList class would support the full List API, allowing it to be used in IDEs or programs where the old List is. My new class would simply be a set of wrappers around the original methods, all very similar to this:

```
public synchronized void removeAll() {
    super.removeAll();
    objectList.removeAllElements();
};
```

Besides creating an OO interface and eliminating redundancy, this code is much safer. The internal Vector is guaranteed not to get out of sync with the List because clients can access it only through the public methods. This achieves a key goal of OO design: An object should guarantee the integrity of its own data. Listings 1 and 2 show two simple classes that represent typical domain objects. OOListTest.java in Listing 3 uses them to demonstrate how

Terminal
robb@quiche:~/temp > java OOListTest
Creating a list of dogs...
Creating the dog list...
Displaying the list...
You chose Spot
You chose Fido
You chose Rex
A list of dogs
Spot who is 5 years old
Rex who is 7 years old
Fido who is 2 years old
Fluffy who is 3 years old

Figure 1. **OOList.**

the new OOList class can be used. Figure 1 shows the results.

Extending the Swing JTable

Swing brought many improvements to Java. The JList class, in fact, allows any object type to be added to its list. This fixes the AWT List's String-only problem, when Swing can be used. There are still some areas of Swing, though, that can use improvement.

I soon found myself in a typical database programmer's situation: I was writing lots of applications where I needed to display information from a database, let the users change some of it, and then store the updates. I had already made an OO API for my data storage (which was relational), but making the UIs was turning into a lot of work. The programs were designed for a biological laboratory. Typically, a user would want to look at a set of 50 or so samples, each one having 10 properties such as temperature or quantity. The user could then edit some of the properties and save the modified objects.

The Swing JTable was the natural choice for displaying this data. The easiest way to store information in a JTable is to use the DefaultTableModel, which accepts a two-dimensional array, or a Vector of Vectors. JavaSoft doesn't recommend this approach, and sure enough, my needs were too complex for the simple DefaultTableModel class. When I realized this, I started to rewrite my code to use the JTable as JavaSoft intended. I made custom TableModel implementations that get plugged into a JTable. The easiest way to do this is to subclass AbstractTableModel and implement the following methods:

- int getColumnCount()
- int getRowCount()
- Object getValueAt(int row, int col)

The JTable will then call these methods when it needs to. Optionally, other methods can be overridden in order to let the JTable change the table data, as well as display more detailed information. Here are some of the more interesting ones:

- boolean isCellEditable(int row, int col)
- void setValueAt(Object, int row, int col)
- String getColumnName(int index)
- Class getColumnClass(int index)

The table model would know about the particular properties of my domain objects, and let the JTable edit and

Feeling Exposed?

Don't Reveal Your Source...

JCloak - Code Obfuscation That Doesn't Byte.

jcloak™

Force5.com

Force5 software

Force 5 and JCloak are trademarks of Force 5 Software, Inc. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries

LISTING 1.

A generic animal class.

```
/**
 * This is a simple abstract class that represents a generic animal.
 * It has three writable properties: age, name, and
 * healthy.
 **/
public abstract class Animal {
    private int         age;
    private String      name;
    private boolean     healthy;

    /**
     * Create a new animal
     **/
    public Animal(int age, String name, boolean healthy) {
  this.age = age;
  this.name = name;
  this.healthy = healthy;
    }

    // "Age" property
    public int getAge() { return age; }
    public void setAge(int i) { age = i; }

    // "Name" property
    public String getName() { return name; }
    public void setName(String s) { name = s; }

    // "Healthy" property
    public boolean isHealthy() { return healthy; }
    public void setHealthy(boolean b) { healthy = b; }

    public String toString() { return name + " who is " + age + " years
old"; }
}
```

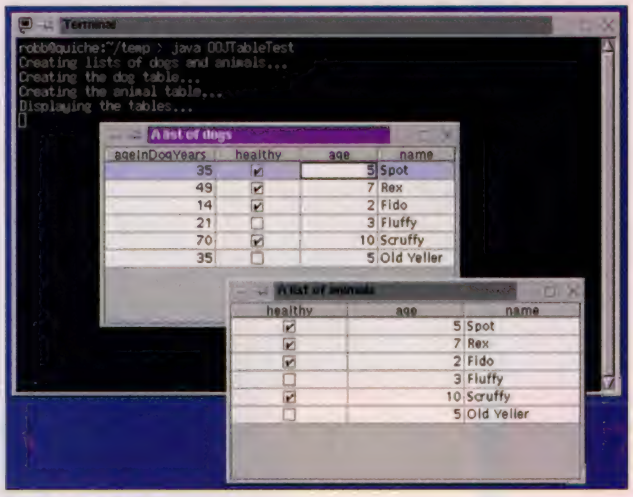


Figure 2. ***OOJTable.***

view them. This worked better—my code was cleaner, but I soon ran into another problem. Every dialog I was writing needed to show a different set of objects, and for each type of object, I found myself making a new TableModel implementation.

It occurred to me that, once again, I was writing somewhat un-OO code in order to work with UI widgets: I had classes that dissembled my objects into scalar values, and reassembled them again. It was turning into a maintenance problem, because changing my domain classes (the actual classes I wanted to work with like "Sample") caused me to

Accelerate your
Java™ development
and deployment

JProbe Performance Tuning Tools
Powerful profiling, memory debugging, thread analysis and code coverage for your Java applications.

JClass Enterprise-ready Java Components
JClass is the world's leading collection of components for developing enterprise-class, data-aware Java interfaces.

Visit KL Group at Booth 505 at Java Business Conference in New York, December 7-9, 1999.
Join us at our educational session,

Directing the Deployment of Enterprise Java Applications

Speaker: Jeff Zado, Product Manager

KL GROUP
Software Development Productivity

KL Group Inc. Toll-Free: 1-800-663-4723 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klgroup.com

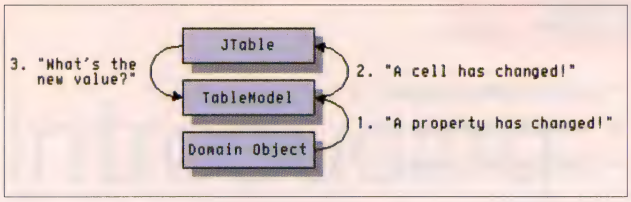


Figure 3. ***Notification.***

change the TableModels that know how to work with them. I started thinking about how I'd like to use JTables if I could. Instead of giving the JTable a two-dimensional array of scalar values, it would be better if I could give it a one-dimensional array of the objects themselves. Objects would be displayed one per row; and the columns would somehow automatically show the objects' properties. But how would this happen? I realized that the JavaBean system would be perfect. The new development process would look something like this:

First, a special TableModel, OOTableModel, is implemented. This class accepts a one-dimensional array of objects in its constructor. It analyzes the objects' class using the JavaBean Introspector to get a list of the class' PropertyDescriptors. Each PropertyDescriptor has information about a single JavaBean property, including:

- A "display name" for the property.
- A "read method" that gives access to the property.
- Possibly a "write method" that sets a new value for the property.

Here's how the basic TableModel interface is implemented:

- getRowCount()—Return the length of the object array.
- getColumnCount()—Return the length of property list.
- getColumnName(i)—Return the display name of property[i].
- getValueAt(row, col)—Get the read method of property[col], and execute it on object[row].
- setValueAt(object, row, col)—Similar to above, but get and execute the property's write method if there is one.

Finally, this last method gives a JTable information about whether a user should be allowed to edit a column:

- isEditable(col)—Get property[i], and return true if it has a write method.

Once the OOTableModel was completed, I needed to make sure that my domain classes adhered to the JavaBean standard. This isn't hard to do: For any property abc, a class should provide access to it via methods named getAbc and setAbc, if it's writable. Alternatively, Beaninfo classes can

LISTING 2.

Dog class extends animal.

```
/**
 * A simple concrete Dog class that extends animal,
 * adding one readonly property: ageInDogYears.
 **/
public class Dog extends Animal {

   /**
    * Create a new dog
    **/
   public Dog(int age, String name, boolean healthy) {
   super(age, name, healthy);
   }

   /**
    * Return the age, adjusted for how long a dog lives.
    **/
   public int getAgeInDogYears() { return getAge() * 7; }
}
```

LISTING 3.

OOListTest.java.

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class OOListTest {

   public static void main(String[] args) throws Exception {
   /*
    * Create a list of Dogs
    */
   System.out.println("Creating a list of dogs...");
   Vector dogList = new Vector();
   dogList.addElement(new Dog(5,  "Spot", true));
   dogList.addElement(new Dog(7,  "Rex", true));
   dogList.addElement(new Dog(2,  "Fido", true));
   dogList.addElement(new Dog(3,  "Fluffy", false));
   dogList.addElement(new Dog(10, "Gini", true));
   dogList.addElement(new Dog(5,  "Old Yeller", true));

   /*
    * Create the list
    */
   System.out.println("Creating the dog list...");
   final OOList list = new OOList(dogList.elements());

   /*
    * Add a listener that retrieves the selected dog
    */
   list.addItemListener(
      new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
            Dog dog = (Dog) list.getSelectedObject();
            System.out.println("You chose "+dog.getName());
      }});

   /*
    * Display the list.
    */
   System.out.println("Displaying the list...");
   Frame f1 = new Frame("A list of dogs");
   f1.add(list);
   f1.setSize(200,100);
   f1.show();
    }
}
```

be generated or written that give detailed information to the Introspector. Animal.java and Dog.java from the previous example are basic JavaBeans. Animal has three writable properties: "age," "name," and "healthy." Dog subclasses Animal and adds a read-only property, "ageInDogYears."

Finally, using the new classes is very easy. Listing 4 (available at *Java Report Online*—www.javareport.com), OOJTableTest.java, gives an example. Figure 2 shows the results. Whenever I have a list of objects that I want to show the user, I instantiate a new OOTableModel with the object list, pass it to a JTable, and display it. The JTable automatically displays the objects one per line, and their properties one per column. The user can click on the cells and change the values. Properties that don't have a set method will be read-only and the JTable won't let the user edit them. It's a bit difficult to see in a screenshot, but the Dog's "ageInDogYears" property is read-only and no edit field appears when the user clicks on it. I also made a subclass of JTable called OOJTable as a convenience to save myself the step of separately instantiating the table model.

Notification

The JTable enhancement described so far is just the beginning of what can be done. One issue that isn't resolved yet is what happens if the objects change while they're being displayed. The TableModel and JTable classes have a nice notification protocol that's easy to use when subclassing AbstractTableModel like I have. The AbstractTableModel has several methods for telling a JTable that something has changed, such as:

- fireTableDataChanged(...)
- fireTableRowsUpdated(...)
- fireTableCellUpdated(...)

The JTable can then query the TableModel for the new values and repaint the changed cells. Figure 3 shows the sequence of notification actions. A domain object lets the TableModel know that it has changed.

Another part of the JavaBean specification describes "bound properties." A JavaBean automatically notifies listeners whenever any of these properties are changed. Using bound properties is like using the Observer/Observable classes, except bound properties are more flexible because the Observable object isn't required to subclass a particular class. Instead, the PropertyChangeSupport class takes care of most of the work involved. Listing 5, Animal2.java, and Listing 6 (both Listings available at *Java Report Online*), Dog2.java, show the modified JavaBeans, which now have bound properties. All that remained was to change OOTableModel in two ways. First, it must try to register itself as a property listener of each object it receives. I did this by using reflection and ignoring any exceptions that are thrown. The TableModel must also be modified to react correctly when it's notified about a change. Its implementation of the propertyChange(PropertyChangeEvent) method acts like a relay, catching the property change notifications and, in turn, sending out notifications about TableModel changes.

Javlin

Introducing the Javlin™

EJB Data Server

from Object Design®

START **Attention Java developers!**

Object Design, the leader in innovative eBusiness data management systems, announces the Javlin EJB Data Server. You use an EBJ server to **build, distribute and scale** your eBusiness logic, so why not do the same for your data?

Javlin gives you a place to store your eBusiness data in the **middle tier** where it's accessed at in-memory speeds. It even synchronizes data server contents with enterprise data sources.

Here's your chance to find out more.

Download a FREE evaluation copy of Javlin

www.objectdesign.com/javlin

Or call us at 1-800-962-9620.

Javlin: Does for data what EJB application servers do for logic.

OBJECT DESIGN®

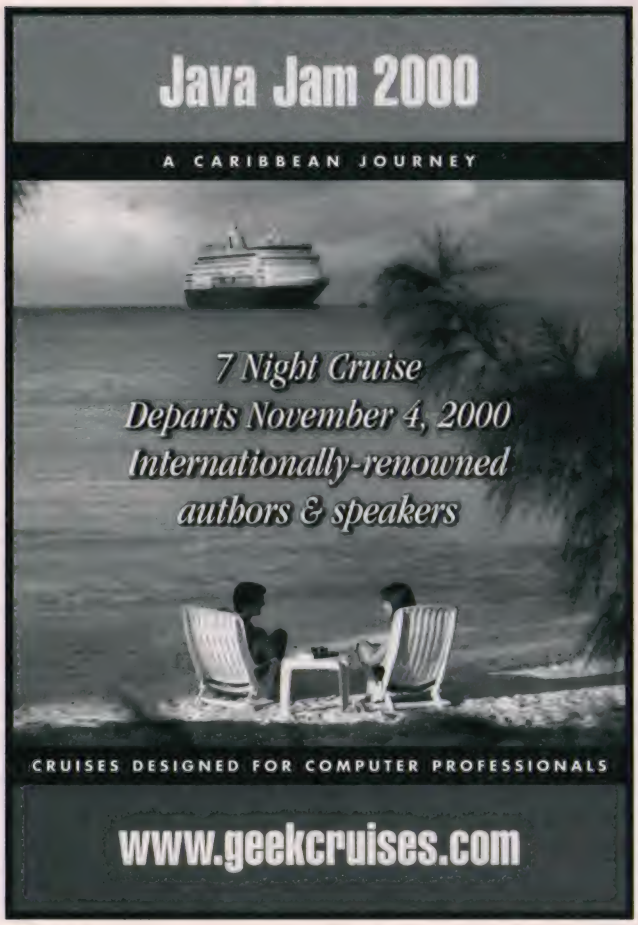
Object Design, Inc. Twenty Five Mall Road, Burlington, MA 01803-4194, email: info@objectdesign.com. Copyright © 1989-1999 by Object Design, Inc. Object Design, ObjectStore, Leadership by Design, ObjectExchange, and the Object Design logo are registered trademarks of Object Design, Inc. Cache-Forward, eXcelon and Javlin are trademarks of Object Design, Inc. All other trademarks are the property of their respective owners.

With these additions, a JTable will automatically repaint itself when the data changes.

Benefits of OO Widgets

I discovered that being able to directly display real objects in UI widgets encourages better and more flexible designs:

- References to many kinds of objects can be passed to the OOTableModel, including persistent database objects, domain objects, report objects, and remote RMI objects. Objects that are cast as abstract classes or interfaces can also be used.
- The system uses standard JavaBean features, which means that most classes don't have to be modified to work with the OOTableModel. Third-party classes can be used without modification or access to their source code.
- The need to spread information about domain objects throughout programs is reduced. Normally, a UI class must know something about the classes it displays. This introduces redundancy, which is a maintenance problem. With OO widgets, objects can be conveniently described just once where they should be—in the class source code. The system then reuses this description at runtime.
- OO design and polymorphism can be used to create interesting effects such as database-like "views" onto the objects. Note the difference between the "Animals" table and the "Dogs" table in Figure 2.
- OOTableModel and OOJTable are consistent with Swing design and can be combined with other Swing features like cell renderers.



Possible Applications

The most obvious use for these new widgets is in easy-to-construct dialogs. Very little extra coding needs to be done aside from checking whether an "OK" or "Cancel" button was clicked. The application can then proceed using the objects like it normally would. Moving beyond this, the notification/update feature has some interesting possibilities. A small system to monitor the state of a running program could be made. Some key objects in the system would be passed to OO widgets, where a user could observe changes that occur in them over the lifetime of the program.

All object references are considered equal and a remote reference can be displayed just as easily. This feature could be used in a distributed application control system. For example, a collection of references to remote servers could be passed to an OOJTable, allowing an administrator to view their status, shut them down, and start them up.

Ideas for the Future

A neat feature of the model–view design is that the pieces can be separated and used in other contexts. The OOTableModel doesn't have to be used with a JTable. Having table-like access to a set of objects and their properties can be helpful in other settings. For example, I've been playing with classes that use a TableModel to generate HTML in servlets. Coupling these with the OOTableModel has made it easy to quickly write Web applications. Taking this further, these HTML producing classes could create a layer between DHTML or HTTP that would change browser events into sets and gets to a TableModel.

An object browser could be created by extending the OO widget idea. For example, maybe clicking the right mouse button over a cell in a JTable would cause that object to be "inspected" in a new window. All of its properties would then be displayed and could be edited. These properties could then themselves be inspected, etc.

The OOTableModel class could be easily modified to accept a Vector or Enumeration in its constructor. This would make it easier to use in typical applications. In these cases, though, there wouldn't be any guarantees that all objects would be the same type, or could be displayed.

Changes in objects can be detected with the notification enhancements I describe. However, changes to the list of objects itself are not accounted for. This might be a handy feature: detecting additions and deletions from the list of objects, and notifying listeners about them.

Finally, it would be interesting to enhance other Swing model interfaces. As I mentioned in the HTML example, a TableModel is a useful tool for abstraction apart from Swing. Having easy-to-use extensions added to the other Swing interfaces could enable new types of flexible applications to be built around them. ■

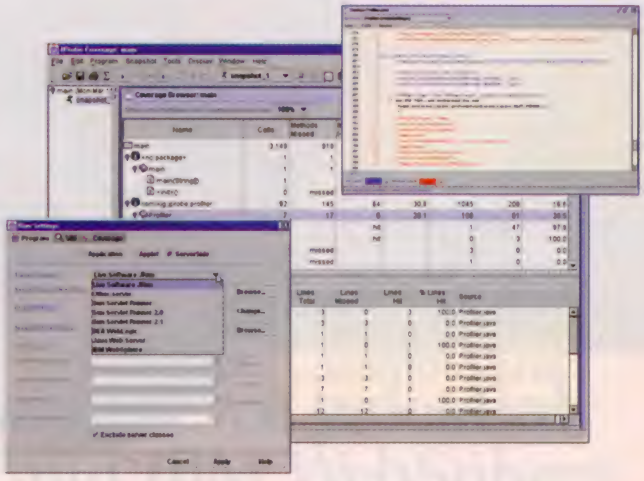
Robb Shecter is a researcher at the German National Research Center for Information Technology (GMD/IPSI) in Darmstadt, Germany. He can be contacted at robb@acm.org.

Quantity reprints of this article can be purchased by phone: 717.399.1900 ext. 139 or by email: sales@rmsreprints.com.

Track down untested code with JProbe Coverage

If you're developing mission-critical Java™ applications, reliability is essential. You need to be sure that your test runs are thorough and accurate, and that untested code won't put your application at risk. Don't let untested code escape you. Track it down with JProbe Coverage, the most accurate code coverage tool available for Java.

Use the easy drill-down navigation and graphical display to identify precisely which lines of code have not been tested. JProbe Coverage gives you all the information you need to ensure your code is covered. Evaluate today!



JProbe Profiler – with Memory Debugger

Debug memory leaks and performance bottlenecks in your Java code

JProbe Threadalyzer

Pinpoint and eliminate deadlocks, stalls, and race conditions

JProbe Coverage

Locate and quantify untested Java code

- Identify and measure untested classes, methods and lines of code
- Zero in on critical code with advanced filters
- Merge total coverage data from multiple runs
- Browse, share & print results in HTML or text
- Analyze server-side code with ServerSide Edition
- Supports JDK 1.1.7, 1.1.8 and Java 2
- Windows 95/98/NT and Solaris versions available
- VisualCafé and VisualAge support

Now with suite-wide Solaris support!

JPROBE Suite

Available in Developer and ServerSide Editions.

www.klgroup.com/track

Introducing JProbe ServerSide Edition!

Web and Application Server Launch Pad:

- IBM WebSphere
- BEA WebLogic
- Allaire JRun
- Sun Servletrunner
- Sun Java Web Server

KL GROUP
Software Development Productivity

KL Group Inc. Toll-Free: 1-888-361-6205 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klgroup.com

© Copyright 1999 by KL Group Inc. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. JProbe, JProbe Profiler, JProbe Memory Debugger, JProbe Coverage and JProbe Threadalyzer are trademarks of KL Group Inc. All other products are trademarks or registered trademarks of their respective companies. JR001

SoftWired iBus

Kurt Vargo/Images.com

Unify the advantages of P&S, Java, and protocol independence

Middleware for the Java Platform

MARCEL ALTHERR, MARTIN ERZBERGER, AND SILVANO MAFFEIS

WE LIVE IN an event-driven world in which businesses need to react quickly to important events if they want to gain or secure competitive advantages. In financial institutions, for example, faster settlement times reduce costs and risks. To achieve this, companies must transform themselves into *real-time enterprises*.[1] A real-time enterprise requires applications and services that are driven by business events, where a business event is any significant change in business data or conditions.

Business information systems are also becoming more heterogeneous, encompassing not only servers, PCs, and workstations, but delivering critical business events also to cellular phones, pagers, and PDAs. The quality-of-service of the communication links needs to be controllable. Certain events need guaranteed delivery and encryption, other events are to be transmitted in a best-effort manner for cost reasons. This is why supporting Middleware must represent this time-critical moment and allow a great deal of flexibility in developing and deploying such systems.

The software bus, or publish-and-subscribe (P&S) paradigm, was invented at the beginning of this decade for modeling event-driven information systems. P&S is a type of messaging Middleware suited for high-performance communication in a one-to-one, one-to-many, and many-to-many fashion. P&S has become very popular in the financial industry, as it allows you to deal with real-time financial data streams in a rather intuitive way: Applications get access to stock quotes and other instruments by simply subscribing to the appropriate information channels of the underlying P&S Middleware.

Nowadays, P&S is also used in such fields as GUI development (Lotus InfoBus), systems integration (TIBCO ActiveEnterprise), real-time information systems (Iona's

Leverage the Web's e-commerce potential with The Object People and IBM WebSphere

Need to make sense of a "web" of technologies?

- Java development
- Enterprise JavaBeans (EJBs)
- Java Database Connectivity (JDBC)
- Remote Method Invocation (RMI)
- CGI, Servlets, Java Server Pages (JSP)
- Server-Side Includes (SSI)
- Applets vs. Servlets
- Secure Web systems

Where does the Web server stop and the application server start? How are all these technologies integrated? How do you make it all work for your business?

The Object People: Your WebSphere Experts

The Object People has a dedicated WebSphere curriculum and experience in delivering WebSphere education to clients in Europe, Asia, Australia and the Americas. We help build e-commerce Internet and Intranet sites for corporate clients in financial, telecommunications, automotive and other industries. And we deliver world-renowned immersion training programs for effective, rapid knowledge transfer.

Get the best of both worlds

- Experienced education, immersion and consulting services from The Object People
- Advanced Web development and management tools from IBM WebSphere

The Object People has the answers to your questions and the solutions to meet your business application needs. We have a decade of leadership and experience in solving problems for mission-critical business applications worldwide.

For details on how The Object People can help you leverage the Web's e-commerce potential, visit our Web site, contact us by e-mail at info@objectpeople.com, or phone one of our service representatives at 613-225-8812.

www.objectpeople.com

CANADA • U.S.A. • U.K. • GERMANY

IBM Business Partner

WebSphere is a trademark of International Business Machines Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Other brand and product names are trademarks or registered trademarks of their respective holders.

THE OBJECT PEOPLE

www.objectpeople.com

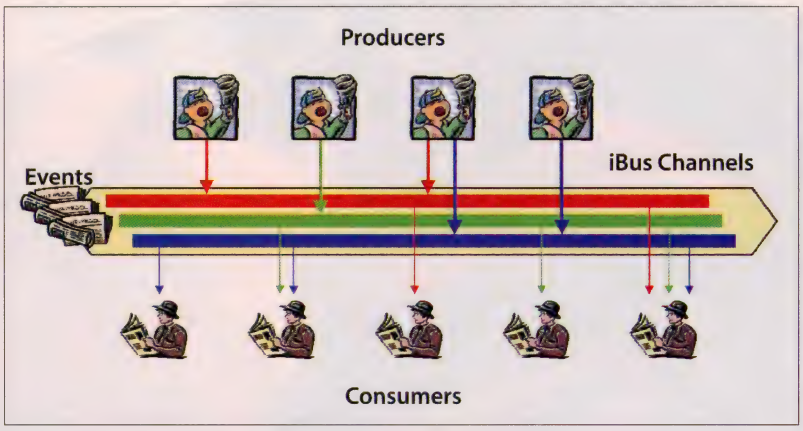


Figure 1. ***Publish-and-subscribe model.***

OrbixTalk), and wide-area event dissemination (Vitria's Velociti).

P&S is of special interest for component systems because the coupling of components can be reduced if the components exchange information via a software bus instead of calling methods on each other directly. This is one of the motivations of the Lotus InfoBus product, for example.

We discuss a new generation of software bus Middleware for the Java platform, called the *SoftWired iBus*. iBus unifies the advantages of P&S, Java, and protocol independence. The goal of iBus is to support the P&S paradigm atop various communication protocols (IP multicast, TCP/IP, Satellite, or Infrared) and with various quality-of-service guarantees (unreliable, reliable, asynchronous, synchronous, encrypted, or persistent communication). iBus is written completely in Java and provides a JavaBeans API as well as a JMS (Java Message Service)-compliant API.

iBus Publish-and-Subscribe API

In the iBus programming model, components exchange events by subscribing to channels and by publishing on channels. The components interested in events are called *consumers*; the components publishing events are called *producers*. A component can be both a consumer and a producer at the same time.

A *channel* can host any number of producers and consumers. When an event is published on a channel, it is transmitted to all consumers via a multicast protocol or by simulating multicast atop a point-to-point protocol. Consumers receive only the events that are published on the channels they are subscribed to (see Fig. 1). iBus channels are named using an URL notation (e.g., "ibus:///datafeeds/quotes/EUR") and a large number of channels can be used simultaneously.

iBus does not demand any network-centric naming service, nor does it make use of object references in the sense of CORBA and RMI. Producers and consumers that use the same channel will find each other spontaneously.

Finally, the *events* transmitted are serializable Java objects. Such objects can represent a business event, an alarm, a JavaBean component, and so forth.

The following classes and interfaces are used when writing P&S applications using the iBus JavaBeans API. The iBus JMS API is not addressed here.

- Channel: class to represent a communication channel by which producers/consumers exchange events. (JavaBean)
- Producer: helper class used to inject objects into a Channel. (JavaBean)
- Subscriber: helper class used to receive events from a Channel and to delegate those events to a local PublishListener object. (JavaBean)
- PublishListener: interface used by consumers for dispatching event objects that arrive on the channels the consumer is interested in.

A typical producer application creates a Publisher object and ties it to an iBus channel by calling the Publisher.setChannel method. Then the producer issues the publish method on the Publisher object by passing an argument of type java.io.Serializable to it. The argument is transmitted to all consumers on the channel by a protocol such as a reliable IP multicast or TCP. As we will later see, various protocols and qualities of service can be associated with an iBus channel.

A typical consumer creates a Subscriber object and ties it to an iBus channel by calling the Subscriber.setChannel method. Note that producers and consumers need to agree on a channel URL in order to interact with each other.

iBus delivers events to consumers by calling the handlePublishEvent method on a listener object that implements interface PublishListener. A listener is tied to an iBus channel by calling the Subscriber.addPublishListener method. The iBus API follows the JavaBeans specification; iBus applications can thus be created within a visual development environment.

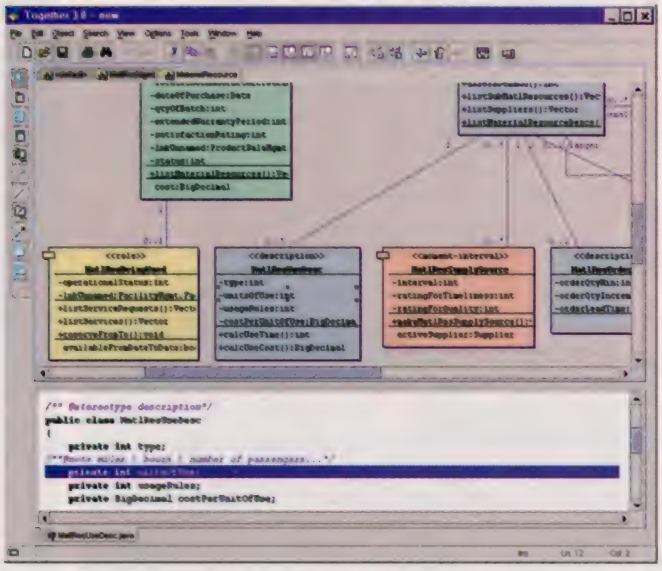
Table 1. iBus P&S beans and their main methods.

Channel	•setChannelURL(ChannelURL) •addChannelViewChangeListener(ChannelViewChangeListener)
Producer	•setChannel(Channel) •publish(java.io.Serializable)
Subscriber	•setChannel(Channel) •addPublishListener(PublishListener)
Requester	•setChannel(Channel) •java.util.Vector request(java.io.Serializable)
Replyer	•setChannel(Channel) •addRequestListener(RequestListener)
<<PublishListener>>	•handlePublishEvent(PublishEvent)

New!

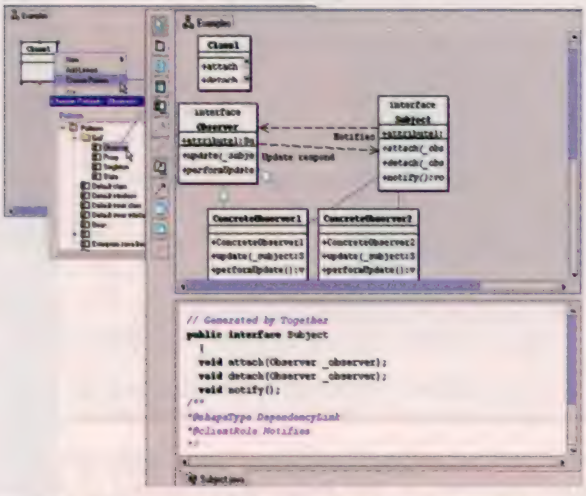
Together/J's EJBs, GoF patterns, and simultaneous round-trip engineering—what Home Depot calls "the backbone of all our Java development"

Download the free TJ Whiteboard and experience TJ for yourself

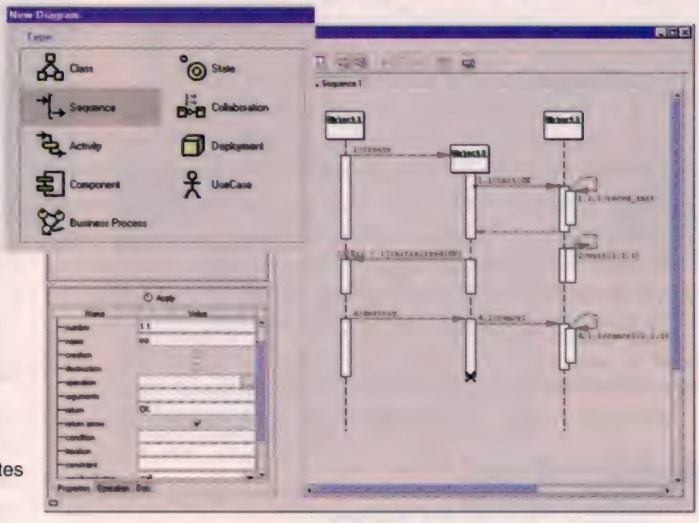


Simultaneous round-trip—always in-sync always up-to-date, complete freedom to work visually or textually as you see fit

Version-control features are a just mouse-click away—complete end-to-end versioning of both models and source across your entire team



Select a class, choose EJB™ or a GoF pattern, and TJ auto-generates it for you – in Java or C++ (or both, in Together/Enterprise)



UML 1.3 (nine kinds of diagrams) including reverse engineering of methods into sequence diagrams. Plus HTML and RTF documentation—invoke it as part of your daily build, deliver always-up-to-date doc to your boss

TogetherJ

Download TJ Whiteboard from www.togethersoft.com/jr/

"Having seen Together/J's direct use of Java code to support the models, I am forced to wonder why anyone would like to work in any other way! I do think that the strength of the product lies in its interoperability — it's a 'complete solution', not just a modeling tool." — Jon Collins, Senior Analyst, **Bloor Research**

"I really like Together/J. No other vender has even come close to making such an intuitive and convenient diagramming interface." —David Marine, Sr. Web Applications Developer, **NetSolve**

"We discovered in our evaluation that other modeling tools only get used during the front-end of a project. Yet we found Together/J to be the one UML modeler that our developers will use from start to finish. Why? It's the only one that delivers simultaneous (rather than batch-mode) round-trip engineering." — Curtis Chambers, **The Home Depot**

"I'm sold... I just finished doing the entire Use Case diagram for the project, (in the format I wanted it), publishing it to a web document on our web site, am working on the sequence and collaboration diagrams right now; and am scheduling a net video discussion group of all our users to discuss the use case designs that we posted... All with Together/J. Your product does verything we need to do now and more..." — Steve McConnell, **The Shaw Group**

"Together/J is an excellent modeling CASE tool for Java developers... It also meets the needs of real-world object modelers. Its support for language-specific features such as inner classes and JavaBeans, puts it miles ahead of other CASE tools that merely support the Java-language syntax... Together/J 3's support for key UML 1.3 diagrams, patterns, and Java-specific language features makes it a valuable addition to your Java development toolset." — **Jolt Award '99**

TogetherSoft L.L.C.
1720 Leigh Drive
Raleigh, NC 27603
(919) 772-9350
www.togethersoft.com

rated jars top 1%

Editors' Choice JavaWorld 1998

Software Development Product Excellence Award

©1999 TogetherSoft L.L.C. All rights reserved.
Together is a registered trademark of TogetherSoft L.L.C.
Java and EJB are trademarks of Sun Microsystems, Inc.

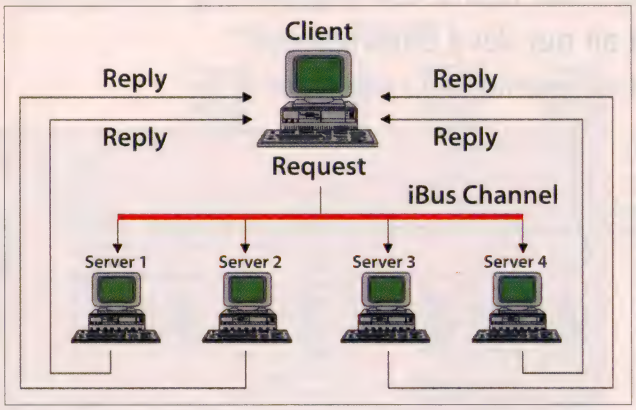


Figure 2. ***Request-and-reply style communication with multiple servers.***

LISTING 1.

EuroProducer.

```
import ch.softwired.ibus.*;

public class EuroProducer {
      public static void main(String [] args) throws Exception {
          Publisher p = new Publisher();
          Channel euroChannel = new Channel();
          ChannelURL url
              = new
ChannelURL("ibus:alias(reliable):///quotes/EUR");
          euroChannel.setChannelURL(url);
          p.setChannel(euroChannel);
          for (;;) p.publish("EUR 10:15AM 0.9156");
      }
}
```

LISTING 2.

EuroConsumer.

```
import ch.softwired.ibus.*;

public class EuroConsumer {
      public static void main(String [] args) throws Exception {
          Subscriber s = new Subscriber();
          Channel euroChannel = new Channel();
          ChannelURL url =
              new ChannelURL("ibus:alias(reliable):///quotes/EUR");
          euroChannel.setChannelURL(url);
          s.setChannel(euroChannel);
          s.addPublishListener(new EuroListener());
          synchronized (s) { s.wait(); }
      }
}

class EuroListener implements PublishListener {
  public void handlePublishEvent(PublishEvent pe) {
      System.out.println("Got exchange rate: " + pe.getObject());
  }
}
```

Listing 1 contains the code of a producer application that transmits an exchange rate for the Euro to its consumers via reliable IP multicast.

Listing 2 contains the code of a consumer receiving the exchange rates and displaying them on the command window.

Any number of producers and consumers can be started and terminated in any order. The model is fully dynamic and allows a producer to be relocated from one machine to another without needing to restart the consumers.

iBus Request-and-Reply API

P&S is an asynchronous one-way communication model used to transmit information from producers to consumers. This is well-suited for event delivery but not adequate for applications where a client queries a server for a piece of information stored in a database. Besides P&S, iBus also supports request-and-reply (R&R) style communication, which works much like CORBA and RMI. The following classes and interfaces are used to write R&R applications:

- Channel: same class used for P&S communication.
- Requester: class that allows a client application to issue a blocking request/reply style operation. (JavaBean)
- Replyer: class that allows a server to answer a request/reply style operation. (JavaBean)
- RequestListener: interface used by a server for dispatching and answering incoming requests.

A typical client application creates a Requester object and ties it to an iBus channel by calling the Requester.setChannel method. Then the client issues the request method on the Requester object by passing a request argument of type java.io.Serializable. The argument is published on the channel and dispatched by the server applications on the channel.

As soon as the first reply is received from a server, the request operation returns control to the client. The return argument is a vector containing the server's reply. A vector object is used because iBus can be configured in such a way that the request operation will block until a reply has been received from each operational server, from a majority thereof, etc. This feature allows the development of fault-tolerant client/server applications and of load-sharing services.

A typical server creates a Replyer object and ties it to an iBus channel by calling the Replyer.setChannel method.

iBus delivers requests to servers by calling the handleRequestEvent method on a listener object that implements interface RequestListener. A listener is tied to an iBus channel by calling the

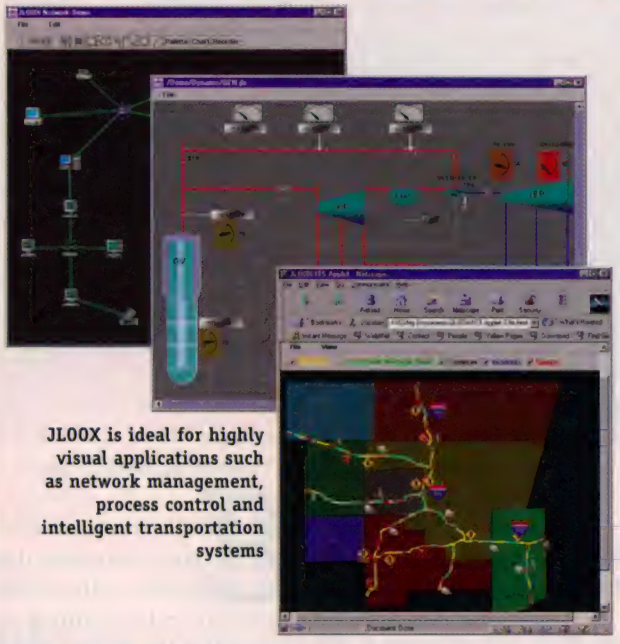
Table 2. iBus R&R beans and their main methods.

Requester	•setChannel(Channel) •java.util.Vector request(java.io.Serializable)
Replyer	•setChannel(Channel) •addRequestListener(RequestListener)
<<RequestListener>>	•handleRequestEvent(RequestEvent)

Accelerate your Java™ Graphics Development Today

JLOOX

JLOOX is the best-in-class graphics toolkit for Java 2 developers. Based on Java2D, JLOOX provides a high-level 100% Java API that offers breakthrough performance and productivity gains for developers of graphics intensive Swing applications. It is ideal for applications requiring highly graphical displays such as networking and telecommunications,intelligent transportation systems, process control and simulation.

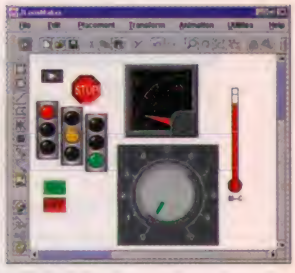


JLOOX is ideal for highly visual applications such as network management, process control and intelligent transportation systems

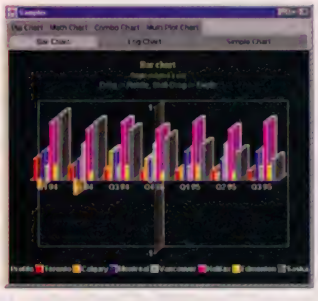
Using the JLOOXMaker interactive editor, complex graphics screens can be created quickly and easily.

Unique advantages:

- Includes an interactive editor for rapid creation and animation of graphic screens
- High-level intuitive API decreases development time
- Offers the rendering power of Java2D
- Pre-built collections of objects provide powerful data-visualization primitives
- Includes the award winning JClass Chart from KL Group
- Can be used with any IDE supporting Java 2
- Applications created with JLOOX have no runtime, royalty or deployment fees



Dynos are powerful data visualization, instrumentation and control objects.



JClass Chart adds powerful graphing capabilities to Java

Accelerating Graphics Development™

LOOX Software Inc.

4962 El Camino Real, Ste. 206, Los Altos, CA 94022 • T:650-903-0942 F:650-903-9824 • www.loox.com

Copyright (c) LOOX Software Inc. 1999. LOOX is a registered trademark and JLOOX is a trademark of LOOX Software Inc. All other trademarks are the property of their respective owners.

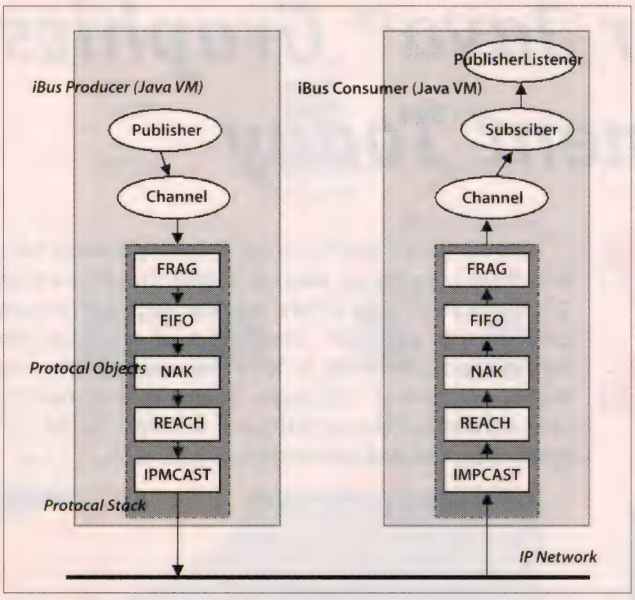


Figure 3. ***A producer and a consumer application communicating via reliable IP multicast protocol stacks.***

Replyer.addPublishListener method.

Listing 3 contains the code of a client application that requests the last rate of the Euro received before market close.

Listing 4 contains the code of a server that maintains a repository with Euro exchange rates and that can look up past rates on request.

Qualities-of-Service

Probably the most distinguishing feature of iBus is its ability to run atop various communication protocols and to support an extensible set of qualities-of-service (QoS). As pointed out, the "classical" application areas of P&S are event-driven business information systems. In our opinion, the P&S paradigm is also well-suited for modeling the exchange of events and information between personal communication devices such as cellular phones and PDAs.

Future information systems will be used to interconnect desktop and embedded devices via on-the-wire and wireless communication links. iBus can support these kinds of applications because the Middleware requires nothing but the JDK, has a small footprint, and can transmit information via virtually any transport medium, be it bidirectional (Intranet) or unidirectional (satellite).

Internally, iBus uses dynamically composable *protocol stacks* for transmitting information via channels. A protocol stack contains one or more *protocol objects* and is used to represent a QoS. iBus comes with protocol objects for reliable IP multicast, TCP/IP, failure detection, data encryption, and so forth. The developer can demand a particular QoS for a channel by creating an appropriate channel URL. The channel URL is then used to create a Channel object, as is shown in the code listings. Note that stacks and protocol objects are invisible to the developer writing iBus-enabled applications but are of interest to the developer wanting to extend iBus with new communication protocols and QoS.

In Figure 3, a producer and a consumer application communicating via reliable IP multicast protocol stacks. FRAG fragments and reassembles messages. FIFO enforces first-in-first-out ordering of messages. NAK implements message-loss recovery using negative acknowledgments. REACH performs failure detection. IPMCAST sends and transmits messages by IP multicast.

iBus channel URLs obey the format:

"ibus:"[QoS:]"//"[address][":"address parameter]/topic

Channel URLs always start with the protocol descriptor "ibus:". The next part is a QoS string indicating the transport protocol and reliability guarantees of the channel.

Optionally, a channel *address* and an *address parameter* can be specified. Thus, an IP address, the wavelength of a satellite link, etc., can be assigned to a channel. The format of the address and the address parameter depend

LISTING 3.

EuroClient.

```
import ch.softwired.ibus.*;
import java.util.Vector;

public class EuroClient {
    public static void main(String [] args) throws Exception {
        Requester r = new Requester();
        Channel euroChannel = new Channel();
        ChannelURL url = new ChannelURL(
            "ibus:alias(reliable):///quotes/EUR/repository");
        euroChannel.setChannelURL(url);
        r.setChannel(euroChannel);
        Vector replies = r.request("Send EUR of last market close");
        if (replies.size() == 0) { /* timeout */};
        Reply firstReply = (Reply)replies.elementAt(0);
        System.out.println("EUR on market close: "
                + firstReply.getObject())
    }
}
```

LISTING 4.

EuroServer.

```
import ch.softwired.ibus.*;

public class EuroServer {
    public static void main(String [] args) throws Exception {
        Replyer s = new Replyer();
        Channel euroChannel = new Channel();
        ChannelURL url = new ChannelURL(
            "ibus:alias(reliable):///quotes/EUR/repository");
        euroChannel.setChannelURL(url);
        s.setChannel(euroChannel);
        s.addRequestListener(new EuroReplyer());
        synchronized (s) { s.wait(); }
    }
}

class EuroReplyer implements RequestListener {
  public void handleRequestEvent(RequestEvent re) {
      System.out.println("Got request: " + re.getObject());
      try { re.getReplyPort().reply("EUR MARKET CLOSE 0.9200"); }
      catch (Exception e) { /* reply operation failed */};
  }
}
```

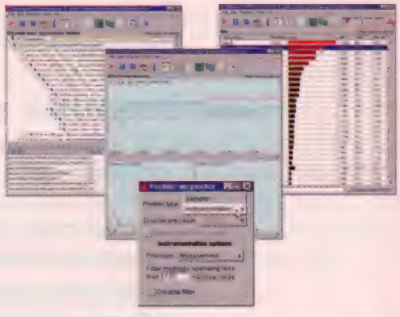
Baaad Code!

Watch Java applications perk up when OptimizeIt™! does the training.

Even well-behaved Java programs can slow down on you. But now you can keep performance problems on a short leash. With OptimizeIt 3.0 Professional.

Thanks to asynchronous data fetching, sampling and advanced profiling technologies, OptimizeIt profiles any large Java environment — web infrastructures, application servers, servlets, Java applications, you name it.

And OptimizeIt doesn't bury you in raw code. Clear, real-time data and



easy-to-read graphs provide the comprehensive and accurate information you need to solve problems — performance bottlenecks, excessive object allocations, memory leaks.

You'll get unique memory leak debugging capabilities with an object reference graph that automatically highlights the actual references causing memory leaks. And a CPU profiler that provides both sampling and instrumentation to tune CPU time measurement to the most efficient level of detail for each test session — with low overhead.

So find Java problems before they become customer problems. Visit www.optimizeit.com today. Download your free trial and see your Java programs jump back to life.

Optimize It!

www.optimizeit.com

OptimizeIt 3.0 Professional supports JDK 1.1x and Java 2.
Now available for Windows, Solaris and Linux!

© 1999 Intuitive Systems, Inc. All rights reserved. OptimizeIt' is a registered trademark of Intuitive Systems, Inc. All registered trademarks belong to their respective manufacturers.

on the QoS. The address parameter can be a port number, a baud rate, etc. If no address is given, then iBus will pick a default. In the case of IP multicast, a multicast address is assigned by computing the hash value of the topic string. This is accomplished without needing to contact any network-centric naming service.

The *topic* is a mandatory string describing the content of the channel. Some examples:

- ibus:alias(reliable):///quotes/EUR: Reliable IP multicast is used. The IP multicast address is computed out of the hash code of the topic.
- ibus:alias(reliable)://251.1.1.13/quotes/EUR: As in the previous example, but a specific IP address is assigned to the channel.
- ibus:CRYPT:alias(reliable):///quotes/EUR: Data is encrypted and sent via IP multicast.
- ibus:CRYPT:TCP:///quotes/EUR: The events are transmitted via TCP instead of IP multicast. This URL can be used by an applet to receive exchange rates nearly in real-time from an Internet server.

Conclusion

More and more businesses need to react to important events immediately and to disseminate such events to many receivers. Furthermore, a component-oriented approach is often used to cope with the inherent complexity of modern information systems. This calls for a new breed of Middleware that is event-oriented and able to adapt to various transport mechanisms and QoS requirements.

Our goal has been to provide this type of Middleware by developing the iBus—a publish-and-subscribe messaging Middleware that is pure Java, has small footprint, and can run atop various IP and non-IP communication protocols. iBus can be used through a JavaBeans API and through a JMS-compliant API. ■

Reference

1. Skeen, D., "Enabling the Real-Time Enterprise," *BYTE Magazine*, Jan. 1998.

Marcel Altherr is CEO and Martin Erzberger and Silvano Maffeis are Executive Vice Presidents of SoftWired Inc. in Zurich, Switzerland. They can be contacted at info@softwired-inc.com.

URLs

Iona's OrbixTalk
www.iona.com/

TIBCO's TIB
www.tibco.com/

Vitria's Velociti
www.vitria.com/

Lotus InfoBus
java.sun.com/beans/infobus

JMS standard
java.sun.com/products/jms

SoftWired's iBus (Evaluation Version Available)
www.JavaMessaging.com/ibus

Quantity reprints of this article can be purchased by phone: 717.399.1900 ext. 139 or by email: sales@rmsreprints.com.

Now in our 9th Successful Year!

Objekt-orientiertes Programmieren

OOP 2000

MÜNCHEN

Conference: January 24–28, 2000
Exhibition: January 25–27, 2000
Internationales Congress Center München

SIGS EXPO for JAVA

Integrating Today's Software Engineering Solutions

Straight Talk From the World's Leading Object and Java Technology Experts

Join 3,000 of your colleagues at what historically has been one of the most respected software development related conferences in the world. For 9 successful years, OOP 2000 has pioneered object based solutions by providing an expansive and fully comprehensive forum for users, experts and leading vendors. You can rely on OOP to maintain its long standing and rich tradition of high quality instruction, innovative ideas and long term benefits.

Get Practical Specifics to Help Plan and Build Quality Software

- Choose from more than 90 sessions with information you can immediately use
- Explore the largest array of object and component based development tools ever gathered in one place
- Get pure intelligence without the hype in a neutral, by-developers, for-developers environment
- More advanced classes, geared to users with 4+ years of OT experience
- Gain clarity on Java programming complexities and be able to build large-scale Java applications
- Learn to deliver enterprise objects over the Internet
- Increase productivity by implementing the UML in your organization
- Learn to really integrate component strategies on the server
- Learn to architect, build and maintain e-commerce systems

Three Day Exhibition 25-27 January, 1999

Over 100 leading vendors will be on hand to demonstrate the latest products. Have your questions answered, take a sneak preview at what's new and discover what works best for you. Plan to attend all the free special events. Check website for details.

What's New in 2000?

- A full track devoted to XML technology
- E-commerce integration and development
- Linux Developers Forum, Solaris Developer Day, Software Quality Day
- A 4-day event- more classes and options than ever before
- Explore the "ins & outs" of Java 2

Plus, what you need to know about:

Swing components, CORBA 3, UML 1.x, InfoBus, COM+, ATL, MTS, Application servers, DOM, XFDL, Python, RMI, Java Plug-in Technology, Jini & JavaSpaces Technology, JavaServer Pages, Servlets, Java HotSpot Virtual Machine & much more!

Customize Your Program

Select from 8 tracks. You have the flexibility to mix and match classes to customize the training to fit your individual needs. Register for 4 days, pre– and post–conference tutorials, and attend night school. Simultaneous tracks focus on today's hottest topic areas:

- Analysis and Design/UML
- Large-scale Application Development
- Power Java
- CORBA and COM
- Advanced Java
- Internet Transaction Processing/e-commerce
- Component Based Development
- XML Technology

What You'll Learn

- Using Design Patterns to Build Application Architectures
- Understanding XML & DOM
- Using Transactional Components
- Strategies for Adopting CORBA for Application Development
- eXtreme Programming Workshop
- Using EJB
- Successfully architecting & Building E-commerce Systems
- CORBA vs. MTS/COM vs. EJB - the middle-tier batter
- Designing flexible Java
- Distributing Components with EJB
- Frameworks and Patterns with UML
- Mapping Objects to Relational Databases
- Integrating Java & XML

Benefit From Our Commitment to Excellence

SIGS Conferences has a 10-year tradition of presenting the world's most comprehensive, vendor-neutral, educational conferences focusing on advanced software technology. Over 20,000 developers from 60+ countries benefit by attending out events annually. Each conference is a unique combination of technical sessions and exhibitions where the pages of JOOP, JAVA REPORT, COMPONENT STRATEGIES, C++ REPORT, APPLICATION DEVELOPMENT ADVISOR, OBJEKTspektrum and JAVA SPEKTRUM come alive for you!

Visit Our Website!

Visit our website at: **www.oopconference.com** for all the latest details on the show. We'll give you a **FREE** issue of OBJEKTSPEKTRUM and JAVA SPEKTRUM just for visiting.

For additional information, please contact:
Phone: +49 (0) 2202-9372-0 • Fax: +49 (0) 2202-9372-2 • Email: infogmbh@sigs.com • URL:www.oopconference.com

Presented by: SIGS CONFERENCES OBJEKTspektrum

Media Sponsors: SIGS Application Development ADVISOR, Java SPEKTRUM

Craig Larman / clarman@acm.org

The Presentation and Domain layers

WE PREVIOUSLY[1] CONSIDERED the partitioning of a business information system into distinct layers and subsystems based on functional cohesion and a desire to establish a separation of concerns in the large-scale software architecture of the system. In addition, we explored the representation of these logical partitions in a physical design with Java packages and associated package naming conventions.

Once the large-scale partitions—such as a presentation and application logic layer—have been chosen, we can focus on a finer scale of design: the detailed design related to a partition. We are applying a top-down decomposition; first in terms of the macro-scale architectural components, then down to the micro-architectures within a partition, and finally, to the design of individual Java classes. This immediate series of columns will focus on smaller-scale design patterns related to individual classes and micro-architectures, whereas in the future, we'll explore design patterns related to the macro-scale and distributed design.

We start by exploring the design of the presentation and application logic layers in greater detail and the relationship between them. We re-examine real use cases, the coupling and communication between the presentation and application logic layers, and consider the question: "What's in a name?"

The Presentation Layer—Some Design Tips

Besides the following tips and the discussion of the coupling between the presentation and other layers, the topic of how to design a good graphical user interface (GUI) is outside the scope of this column. I recommend employing a GUI-design expert for windows style and navigation. Although not profoundly complicated, this is a non-trivial subject that an experienced eye greatly contributes to.

Craig Larman is Director of Process and Methodology at Valtech in Dallas, the author of *Applying UML and Patterns—An Introduction to Object-Oriented Analysis and Design*, and *The Java 2 Performance and Idiom Guide*.

Windows Avoid Application Logic Responsibilities

In general, it is desirable to not embed application logic responsibilities (e.g., calculating the sales tax for a video rental) within the methods of GUI windows (e.g., subclasses of JFrame). Rather, the presentation layer receives requests to do work from external actors and then delegates the work to the application logic layer. This allows different presentations/windows to be attached to application logic—which increases flexibility and reuse potential and allows the application logic layer to be physically distributed onto another compute node. A subsequent section discusses the connection of this layer to the application logic layer.

Real Use Cases

Essential and real use cases were covered in a previous column.[2] Because the creation of real use cases occurs during the design phase and strongly relates to the presentation layer, a short review is in order. A real use case is a design-oriented description of a use case that includes commitment to specific technologies in the realization of the use case. Concretely, a real use case may discuss the details of input and output technologies, such as GUIs with screen shots and navigation. In contrast, an *essential* use case, which is more analysis-oriented, abstracts beyond ephemeral and variable technologies that can be used in the current design of a use case.

Following is an example of part of the Log In use case expressed in a relatively essential form; it will have been written much earlier in the analysis phase.

Actor Action	System Response
1. This use case begins when the Clerk identifies himself/herself.	2. Presents options.
3. ...	4. ...

Find hidden problems automatically with NuMega™ DevPartner™ for Java!

Serious problems can lurk quietly in your Java™ code, and if left undetected, can cause performance and reliability issues after your software is deployed. **NuMega™ DevPartner™ for Java** from Compuware® automatically finds performance bottlenecks and hidden runtime errors in your Java code, which can save you days of valuable development time.

DevPartner for Java is a suite of developer productivity tools that accelerate software development and improve the performance and reliability of your Java code. DevPartner for Java includes **TrueTime** — the award-winning performance analyzer that automatically pinpoints performance bottlenecks, with or without source code; **TrueCoverage** — the code coverage analyzer that automatically measures and tracks tested code, untested code and code base stability; and **JCheck** — the visual thread and event analyzer that automatically detects thread errors, runtime errors, leaks, unhandled exceptions and other potentially serious problems. Plus, all DevPartner for Java tools have powerful multi-VM switching capability, which can help you find behavioral problems that occur when your code runs in different virtual machines.

Find hidden problems in your Java code with NuMega DevPartner for Java, and deliver the fastest, most reliable Java applications and components with confidence!

Supports Java 2

Accelerating Software Development
with
NuMega DevPartner for Java from Compuware

NuMega DevPartner for Java includes:

NuMega TrueTime™ Java Edition — Automatic Performance Analysis and Optimization
NuMega TrueCoverage™ Java Edition — Automatic Code Coverage Analysis
NuMega JCheck™ — Visual Thread and Event Analysis

Call today to request your copy of the technical white paper
"Effective Unit Testing for Java Developers"
or download the paper now by visiting
www.compuware.com/numega

1-800-4-NUMEGA
www.compuware.com/numega
30-Day Money Back Guarantee

JAVA DEVELOPER'S JOURNAL JDJ WORLD CLASS AWARD

COMPUWARE
People and software for business applications™

Copyright © 1999 Compuware Corporation. All rights reserved. All trademarks and registered trademarks are the property of their respective owners.
Java and all Java-based marks are trademarks of Sun Microsystems, Inc. in the United States and other countries.

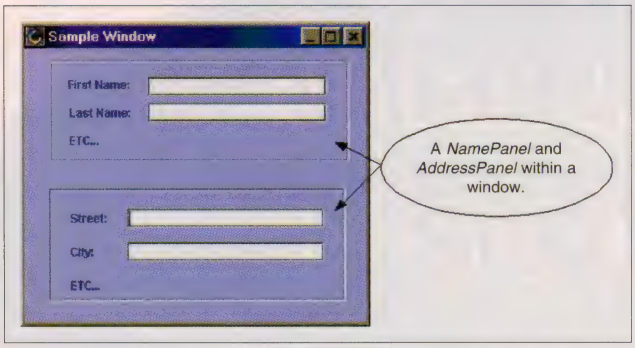


Figure 1. ***Using reusable subpanels to compose a window.***

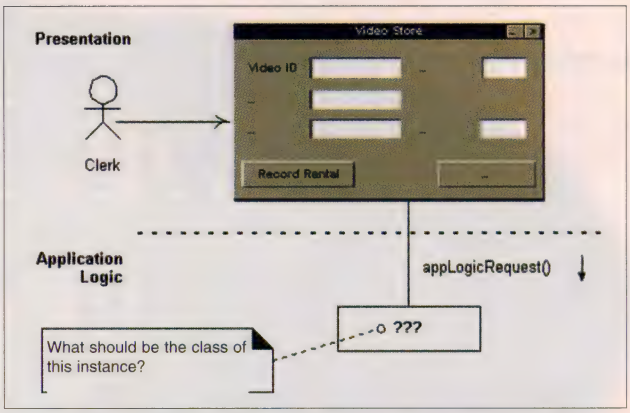


Figure 2. ***What object in the application logic layer should receive requests?***

Here is the same Log In use case expressed in a truer design-oriented form; its writing may be deferred until the design of the presentation layer is considered in detail.

Actor Action	System Response
1. This use case begins when the Clerk types in his/her ID and password in Dialog Box A, in the provided text boxes, and presses the "Log In" button with a mouse.	2. Displays options via the toolbar shown in screen shot 6.
3. ...	4. ...

As this example suggests, if there is a GUI involved, the real use case can include screen shots and detailed discussion of the interaction. More broadly, during the design phase, it is useful to define at least a rough outline of the windows and navigation between them—the storyboarding of the user interface (UI). This UI storyboarding can be recorded in real use cases using a combination of classic use case narrative, screen shots, and directed graphs that diagrammatically summarize the navigation between windows.

Reusable Subpanels

It is desirable to create reusable GUI components for common elements, such as to edit name and address information. In Java, this can be accomplished by creating subclasses of JPanel that contain related widgets. For example, we can create a generic and reusable `AddressPanel` and `NamePanel` and then compose them within many windows (Figure 1). In addition, each panel should expose (as a JavaBeans property) its associated information object. An `AddressPanel` will have a `getAddress()` method that returns an Address instance, and so on.

Connecting the Presentation to the Application Logic or Domain Layer

The vernacular of object technology refines the general notion of "application logic."[1] It is common to place application logic responsibilities into software types whose names correspond to familiar concepts in the domain (e.g., "VideoRental"). These are commonly known as *domain types* or *business objects*. Hence, it is common to speak of the *domain layer*, the collection of types that contain methods with application logic and whose names and responsibilities reflect the core business terminology and logic. In addition, it is not uncommon to factor out special application coordination or control behavior into a separate family of types; this is known as the *application (or transaction) coordination* layer.

This leads to a very interesting and central design question in object-oriented (OO) design: When the application logic layer receives a request for work from the presentation layer (via a message), what objects should handle the request? At one extreme we could have one class—`TheBigClassThatDoesEverything`—receive all requests and handle all the work in its methods. At the other end of the spectrum, dozens or hundreds of objects may collaborate to fulfill the request. Which design is better, and when and why? If many classes of objects are involved, what are their names and responsibilities, and how exactly do they collaborate? These are some of the key questions in the OO design of the application logic or domain layer.

Sending Requests to the Application Logic Layer

However, before diving into these questions in greater detail, it is worthwhile to resolve the following question:

When an object in the presentation layer wants to send messages "down" into the application logic layer, to which object in the application logic layer should it send the message? In other words, who starts the ball rolling down in the application logic layer? (Figure 2).

The GRASP Controller Pattern

There does exist a family of very fundamental principles or common practices in OO design and the assignment of responsibilities to objects, such as design to support low coupling, high cohesion, and so on. To aid in learning and memorization, I've organized these into a family of principles or patterns named the GRASP patterns—General Responsibility Assignment Software Patterns, which are discussed extensively in Applying UML and Patterns.[3] Although the pattern names and their organization into the GRASP patterns are not very important except as a learning aid, the underlying principles that the GRASP patterns try to communicate are important.

One of the GRASP patterns answers the common question posed in Figure 2: What object in the application logic layer should receive a request from the presentation layer?

Work Smarter.

TowerJ™ 3

Mix-Mode Java Deployment Solutions

Industry analysts and independent benchmarks agree, TowerJ 3 is becoming the lead dog for deploying business-critical server-side Java™ applications. With its innovative mix-mode deployment technology, TowerJ is helping business around the world to create dynamic applications that take advantage of both standard byte code and optimized pre-compiled native code. This is allowing these top dog companies to significantly improve the price/performance of their enterprise servers while enhancing the security and control of their applications. If you are developing and deploying business-critical server-side Java applications, take TowerJ for a spin, and see how getting even more value from your Java can separate you from the rest of the pack.

For more information and a free evaluation go to
www.towerj.com/jr

Teach your Java a new trick.

TOWER TECHNOLOGY

This is the Controller pattern, so called because the object that receives the request for work has sometimes been called the "controller" of the request in that it is responsible for controlling or coordinating the handling of the request. Note that the "controller" object does not necessarily (or usually) do the work—it is likely to delegate to other objects.

The Controller pattern, which expresses common choices among object designers, states that one of the following objects is commonly chosen to receive a request from the presentation layer (or more generically, the input handling layer):

1. An object whose class name suggests a kind of large "covering" on something, such as a name reflecting the overall business domain. For example, the class name VideoStore. This is an example of a kind of Façade over the application logic layer. A Façade is one of the Gang of Four design patterns[4] that suggests limiting coupling between different subsystems (or layers) by funneling all messaging into a subsystem via one object—the Façade.
2. An object whose class name suggests the current use case within which these requests occur. For example, many video store requests occur within the Rent Items use case, so we could create a Java class named RentItemsUseCase or RentItemsHandler. In this approach, all requests related to one use case are received by an instance of the use case object.

These options are illustrated in Figure 3. The choices presented by the Controller pattern are not exhaustive—other choices are appropriate in certain circumstances. Nevertheless, these are common.

There are other issues to consider in communication between the presentation and domain layer, such as how to communicate from the domain layer "up" to the presentation layer while still maintaining a low-coupled design.

What's in a Name? Chunking, Semantic Gap, and Naming Things

As illustrated in the previous Controller pattern example, it is common to create software classes whose names reflect the domain vocabulary, such as VideoStore, RentItemsUseCase, Rental, and so on. In other words, to create Java classes whose names correspond to some of the names of concepts in the conceptual model. It is so very common to create software classes with such names that we may not have ever reflected on why this is so. Why do books and educators discussing OO design commonly show the use of classes whose names reflect domain vocabulary? There are actually some fairly deep issues here.

One issue relates to object technology as a packaging or chunking mechanism for program code. We wish to organize code into manageable units rather than one giant routine, to improve clarity, ease of modification, reuse, and so on. Various kinds of units for packaging code exist—most commonly regular procedures and functions in languages such as C, and into methods within classes in OO languages such as Java. Practitioners of object technology claim—I among them—that factoring and packaging into classes that bundle together functionally cohesive behaviors with the data to support these behaviors is a strategy of code chunking that scales up well.

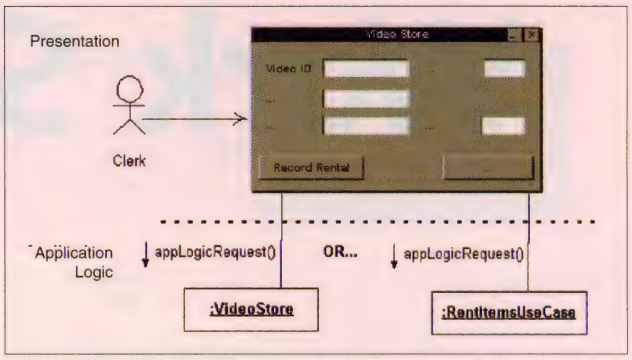


Figure 3. ***Applying the*** Controller ***pattern.***

Once you've decided to chunk code in classes, then why choose a class name such as Rental, and what does Rental do?

Simply, choosing names that reflect the domain vocabulary (e.g., Rental) enhances quick comprehension and provides a clue as to what to expect from the chunk of code in a Rental class. We have a mental or conceptual model of the domain in question (e.g., a video store). In the real world we know that a rental has a due date. Consequently, if we create a Java class named Rental, and give it the responsibility of knowing about a real video rental and its due date, then the Java class Rental somewhat corresponds to our mental model of the real world; that is, it appeals to our "intuitions" about the domain.

This relates to the issue of semantic gap—the gap between our mental model of the domain and its representation in software. At one extreme, we could directly program the video store application in raw binary code to invoke the processor instruction set. We understand that the gap in representations is huge and there will be a real cost—albeit hard to quantify to management—in software with such a large semantic gap because it is hard to comprehend or relate to the problem domain. Closer to the other end of the spectrum is object technology, in which we can chunk code into classes whose names reflect the kind of chunking we perceive in the domain. In the real world, we perceive a "chunk" called a rental, so in software land we have a class called Rental. This closer one-to-one mapping between the domain vocabulary and our software vocabulary and its chunking reduces the semantic gap. This speeds comprehension of existing code (because it works in ways we expect, knowing the domain) and suggests "natural" ways to extend the code in ways that similarly correspond to the domain or appeal to our intuitions of the domain. Put simply, the software model reminds us of the conceptual or mental model and works in predictable ways.

There is a real practical advantage to software models that reduce the semantic gap. I think most software engineers agree this is true, even if it is hard to quantify. Indeed, a proof of this is that Java obfuscators make code hard to understand and inhibit illegal copying by changing the names of classes and methods so that they no longer appeal to our intuitions of the domain. However, if your organization is thinking of adopting object technologies such as Java, and the CEO asks why bother, you can't say, "because it will reduce the semantic gap between our conceptualization of the domain and its soft-

Continued on page 82

Your customers don't like to wait!

(neither do bosses)

DashO™ PRO

REDUCE JAVA APPLET/APPLICATION SIZE UP TO 70%!

3 PRODUCTS IN 1!

Decrease Download Times
DashO-Pro analyzes your program and extracts only the classes your application needs! It then removes unused methods and fields for unmatched space savings!

Obsfucate
DashO-Pro includes the patent pending Overload-Induction™ renaming system. No renamer saves more space or protects your code better!

Optimize
DashO-Pro actually optimizes your code for top performance! **Up to 30% faster!**

preEmptive solutions

www.preemptive.com
1-800-996-4556

©1999 preEmptive Solutions, Incorporated. All Rights Reserved. Java is a trademark of Sun Microsystems. All trademarks are owned by their respective companies.

Steven W. Disbrow / stevendisbrow@ego-systems.com

Building better email with JavaScript

RECENTLY, A FORMER student of mine contacted me for some advice. Was it possible, she wondered, to create a Web page that would email the contents of a form to her and then redirect the client to a "thank you" page? The catch was, it all had to be done on the client side, with no server-side utilities involved, and most importantly, the resulting emailed data had to look *good* (i.e., something that her boss could appreciate).

Making the Email Pretty

This last requirement was going to be the most difficult to achieve. If you've ever used a Web form to send email without using a server-side helper, you know the email the Web browser builds looks absolutely horrible. Depending on the parameters you set in the form itself, the data emailed ends up looking either like a long URL string or divided up by hex codes and lots of unnecessary white space. This, of course, is why you use a server-side application: It takes the form data that it receives, makes it pretty, and then emails the pretty version to the recipient. So, my student wanted to know, could JavaScript be used to get the same effect without using a server-side process?

Well, of course it can, but you have to be a little bit tricky about it. Instead of using a single form, you need to use two forms! The first form is used to collect the data you want to email. The second form contains a submit button along with a hidden field (see Listing 1). When the user clicks the submit button, a JavaScript onSubmit handler function is called. This function takes the data from the first form and copies it into the hidden field in the second form. During this process, we add the necessary formatting to make the resulting email look much nicer. In this example, I'm just using simple new line characters ("\n") to make things nicer. The result is that this hidden field will then contain a nicely formatted block of text that will be emailed when the second form is submitted. And, because there is only one field in this hidden form, there aren't any other fields to clutter up the email.

Steven Disbrow is the owner of EGO Systems, a computer consulting firm in Chattanooga, TN.

Jumping to the Next Page

After we've got our email ready to go, we have to send it and then move on to the "thank you" HTML page. This seems easy enough: Just submit the form using the `submit()` method and then load the new page with a simple assignment to `document.location.href`. Unfortunately, there are some problems with this approach.

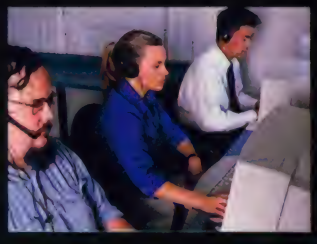
In Internet Explorer (IE), executing the `submit()` method for a form seems to bring execution of the current JavaScript function to a dead halt. So, once the email is sent, nothing else happens! The solution to this is to use the `setTimeout()` method to delay the jump to the other page until one or two seconds after the email is sent. Also, instead of using the form's `submit()` method, we simply return a true from our onSubmit handler to cause the email to be sent. So, the jump to the next page will still take place after the email has been submitted.

Netscape suffers from a much more insidious problem. In IE, when you submit a form that sends an email, the browser simply builds the email and sticks it in your outbox for your email client to handle later on. (Assuming the client is Outlook or Outlook Express.) Netscape, on the other hand, insists on immediately contacting your mail server, sending the email and waiting for a confirmation that the email was sent! The problem is, if anything disturbs this process, like, for instance, jumping to another Web page, the process is terminated, the email is lost, and neither the sender nor the receiver ever knows!

The solution to this problem is much trickier. Even if you can assume that the mail server is available, you simply can't know how speedy it will be at any given moment. So, instead of jumping to another page, we use Dynamic HTML (DHTML) and layers to hide and then display our "thank you" page in the same window we used to gather our data. So, when the user clicks the

THE BEST OF
LIVE AND COMPUTER
BASED TRAINING

View the material anytime
and at any pace



Direct phone access
to Java experts

Peer-to-Peer IDE
for concurrent Teacher/
Student programming

Other Benefits

Cost/Time Efficient

More individual attention
than traditional classroom
or cd-based training

Follow-up help available
after course completion

For a limited time get a **free CD** of Gerry Seidman's lecture on Java Technology and Integrating Java Technology into your Enterprise. To receive your CD, visit our web-site at www.iamx.com for more information.

IAM Consulting
200 West 72nd Street, Ste 35
New York, NY 10023
(212) 580-2700
www.iamx.com

Learn or Improve your Java™ Skills

Unique Distance Training Format

Affordably train 1–10–100

At the office or at home

By one of the top Java technologists

At your own pace and schedule

Not just syntax
we show you how to produce

Clear
Clean
Maintainable Systems

That Run
and...
Produce them in a
reasonable amount
of time

Now for the first time,
courses by Gerry Seidman,
Java expert and pioneer,
are widely available.

IAM Courses Include

Java For Programmers

This applied OO/AD approach to learning Java is ideal for programmers with any background. In addition to its comprehensive coverage of the Java language and libraries, you will learn how to design and implement two and three tier-client/server and Web-based applications.

Expert Java: Designing Enterprise Systems

Designing scalable, secure, distributed applications starts with an understanding of Java internals. From there, learn the technology and design issues behind sophisticated applications using Distributed Objects (RMI/CORBA), framework architectures (Java Beans/Servlets), Legacy Systems (JDBC/JNI), advanced Networking/Multithreading techniques (building resource pooled servers) and much more.

EJB and Transaction Based Systems

Learn how to rapidly design, develop and deploy enterprise wide, server-side business solutions in Java using EJB. Included is a comprehensive coverage of the Java Transaction API (JTS), Java Naming and Directory Services (JNDI), RMI and RMI/IIOP, bundling and deploying EJBs in real application servers as well as practical design techniques for using EJBs in applets, applications and servlets.

Training and consulting at your site is also available.

Java is a registered trademark of Sun Microsystems.

iam

LISTING 1.

getInfo.htm.

```
<HTML>
<HEAD>
<TITLE>Nicer Emails With JavaScript</TITLE>
<script language="javascript">
// This function takes the data from the first form
// and assembles a 'nice' version of it in our
// second form. We then submit the second form
// so that the email received is much nicer to look at.
function doSubmit(iForm, oForm) {
  // Make a reference to the 'assembly' field
  var oAddr = oForm.Address;

  // Start it out with a blank line, for a nice look
  oAddr.value = "\n";

  // Copy the individual form fields from the first
  // form to the second. Put a newline at the end of
  // each field so they look nice.
  oAddr.value += iForm.fname.value + "\n";
  oAddr.value += iForm.lname.value + "\n";
  oAddr.value += iForm.city.value + "\n";
  oAddr.value += iForm.state.value + "\n";
  oAddr.value += iForm.zip.value + "\n";
  oAddr.value += iForm.comments.value;

  // Now it's time to go to the confirmation page and send
  // the email. However, if we send the email first, the new
  // page won't load. And, if we load the new page, we lose this
  // page, so we can't send the email. So, we use the
  // setTimeout method to delay the loading of the new page until
  // AFTER the email has been sent.
  // Of course, this has to be done differently in both Netscape and
  // Internet Explorer. So, first we see which browser is running...
  if (navigator.appVersion.indexOf("MSIE") > -1) {
      // do things the IE way
      setTimeout("document.location.href='thanks.htm'", 2000);
      // Returning true will send the email
      return true;
      }
  else {
      // do things the Netscape way
      oForm.submit();
      document.layers["main"].visibility = "hide"
      document.layers["thanks"].visibility = "show"
      }
  }
</script>
</HEAD>

<BODY>
<LAYER name="main" VISIBILITY="SHOW" WIDTH="100%"
HEIGHT="100%">
<!— This form is where we gather the data —>
<form name="getData">
  Please Tell Us How To Contact You:<br>
  First name:<input type="text" name="fname"><br>
  Last name:<input type="text" name="lname"><br>
  Address: <input type="text" name="addr"><br>
  City: <input type="text" name="city"><br>
  State: <input type="text" name="state"><br>
  Zip: <input type="text" name="zip"><br>
  Comments:<br>
  <textarea name="comments" rows=10 cols=40></textarea><br>
</form>
<!—
This second form is where we build a 'nice' version of the data that
we want to email.

Notice that our 'submit' button is in THIS form, not the other one.
This is so we can use an onSubmit handler to work around a
Netscape 'feature'.
—>
<form name="sendData"
  onSubmit="doSubmit(document.getData, document.sendData)"
  action="mailto:user@domain.com"
  method="POST"
  enctype="multipart/form-data">
  <input type="hidden" name="Address" value="">
  <input type="submit" value="Send Info">
</form>
</LAYER>
<!— The DIV Tag lets us hide this in Internet Explorer —>
<DIV ID="thanks" style="background-color:transparent; visibility:
hidden">
<!— while the LAYER tag allows easy manipulation via Netscape —>
<!— Put your thank you message between the following layer tags
—>
<LAYER name="thanks" VISIBILITY="HIDE" WIDTH="100%"
HEIGHT="100%" PAGEY="0">
<center>
<p>
<b>Thanks for the info!</b>
</center>
</LAYER>
</DIV>
</BODY>
</HTML>
```

submit button, we build the email and submit the form. Then, we use DHTML to hide the layer containing all the form elements and reveal the layer containing our "thank you" message. At this point, even though a new "page" is being displayed, the old page is still there, making its connection with the mail server and sending the mail.

Putting It All Together

Listing 1 shows this code in action. When the user clicks the submit button, the doSubmit() function is called. This function takes all of the fields from the first form and places them, separated by new line characters, into the hidden field in the second form. After this is done, the function determines which browser is running.

If IE is running, a call to the setTimeout() method is used to queue up a jump to the thanks.htm file (shown in Listing 2) two seconds later. The function then returns a true value, causing the browser to actually build and send the email.

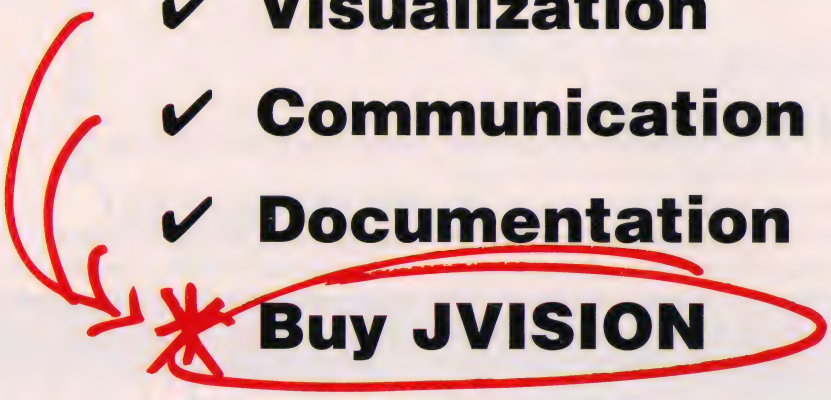
If Netscape is running, the function calls the submit method for the second form, beginning the process of contacting the mail server and sending the email. Then, the function hides the layer containing the form elements and displays the layer containing the "thank you" message.

DHTML Considerations

Examining Listing 1, you'll notice the presence of both <DIV> and <LAYER> tags. The <DIV> tags are used to hide the "thank you" layer from IE users, and the <LAYER> tags are there to allow Netscape to show and hide the appropriate layers. Theoretically, you should only have to use the <DIV> tags. (Both

Continue on page 82

Programming complex projects requires:



- ✔ Visualization
- ✔ Communication
- ✔ Documentation
- ✱ Buy JVISION

JVISION™

$99 or 30 day try before you buy

OTJ™

Object Technology for Java™

Free Download

Visualize your architecture and document projects effortlessly with instant UML diagrams from Java™

Visual Café™ integration add-on

Documentor™ add-on product allows HTML export and instant website creation for multiple diagrams, plus JavaDoc™ integration

products@object-insight.com

Training:

Understanding Objects
Using UML with Java™
Designing Patterns for Java™
OO Software Development
Introduction to Java™
Java™ Basics
Using IBM VisualAge™ for Java™
Java™ and the Internet
Java™ Technology Forecast

training@object-insight.com

OBJECT™ insight

www.object-insight.com

Java™ and JavaDoc™ are trademarks of Sun Microsystems. VisualAge™ is a trademark of IBM. Visual Café™ is a trademark of Symantec. Object Insight™, JVISION™, Documentor™ and OTJ™ are trademarks of Object Insight, Inc.

Kevlin Henney / khenney@qatraining.com

Something for nothing

IT HAS BEEN said that in the days of punched cards, if you wanted to find out how heavy a piece of software was you weighed all the holes. In spite of its apparent lack of substance, there are features of software that make it more substantial to its users and developers: You can develop classes, deploy them, reuse them, etc.; you can run applications and interact with them; you can describe software—especially software built with components and running with objects—in very physical terms.

If you want something really insubstantial you have to scratch the software development surface a little deeper: What are the successful design structures, concepts, and practices that go into making up a system? How do you communicate a piece of design to a colleague or express your ideas in code? What are the benefits of one solution over another, and how does applying a solution affect the subsequent design? For software, patterns seem to offer a discipline based on applied thought; naming, structuring, and communicating recurring solutions to problems.

In software development, the most visible popularization of patterns is the Gang of Four's (GoF) *Design Patterns* catalog,[1] containing 23 common design patterns found in object-oriented (OO) systems. However, these are not the only patterns, and there is a wealth of literature produced by the patterns community that focuses on patterns at all levels and across many different domains. This column is going to look beyond the relatively familiar GoF territory, presenting other examples that demonstrate patterns and pattern concepts of use to the Java developer, as well as some revisits of GoF patterns from a different perspective.

So, without further ado, let's look at a useful tactical pattern: Null Object[2] is perhaps best described as something based on nothing.

Kevlin Henney is a Principal Technologist with QA Training in the UK.

Null Object

The intent of a Null Object is to encapsulate the absence of an object by providing a substitutable object that offers suitable default, do nothing behavior. In short, a design where, as William Shakespeare's King Lear would say, "nothing will come of nothing."

Problem

Given that an object reference may optionally be null, and that the result of a null check is to do nothing or use a default value, how can the presence of a null be treated transparently?

Example

Consider providing a logging facility for some kind of simple server service that can be used to record the outcome of operations and any significant events in the operation of the service. One can imagine many different kinds of log, such as a log that writes directly to the console or one that uses RMI to log a message on a remote server. However, a server is not required to use a log, so the association between the server and log is optional. Figure 1 shows the relationships diagrammed in UML, and Listing 1 shows the Log interface and two possible simple implementing classes.

In Listing 2, we can see that in using a Log object we must always ensure that we first check it is not null, because we know it is optional, i.e., logging need not be enabled for a service.

Forces

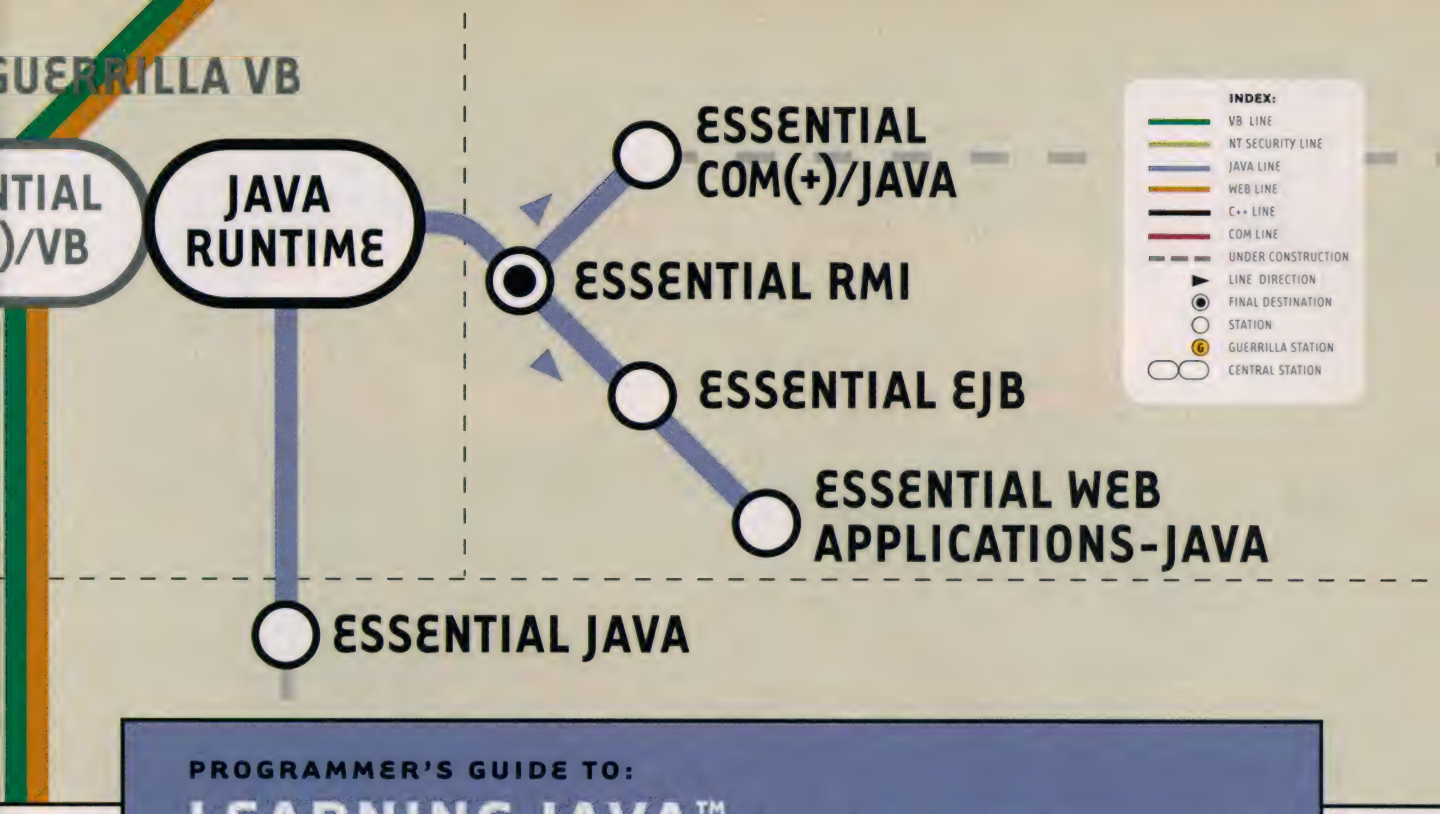
There is a great deal of procedural clunkiness in code, such as:

```
if(log != null)
  log.write(message);
```

This can be repetitive, as well as error prone: It is possible to forget to write such a null guard, and repeated checks for null references can clutter and obfuscate your code.

The user is required to detect the condition and

*Complete references are available at *Java Report Online*—www.javareport.com



PROGRAMMER'S GUIDE TO:

LEARNING JAVA™ for distributed computing

FASTEST TRACKS

ESSENTIAL JAVA

This course is for professional developers currently working in another language who want a fast-paced and in-depth introduction into using Java so that they can create stand-alone applications immediately.

INTENSIVE JAVA

This course is about getting inside the Java virtual machine to better understand its expectations, logic, design and resulting behavior with your application. You'll learn many undocumented tricks and techniques that the VM uses and how you can surmount any optimization, deployment, and security problems in your Java application.

ESSENTIAL RMI

This course is about the technology behind building distributed systems with Java and the core RMI features in the JDK. You will be provided with a real understanding of how to get each feature to work for you and the strengths and weaknesses of each.

TRANSACTIONAL PROGRAMMING WITH EJB

This course will show you the programming model defined by the EJB's specification and builds on your knowledge of Java RMI.

ESSENTIAL WEB APPS-JAVA

This course focuses on writing distributed applications written in Java that use browser-based clients. You'll learn about Java applets and servlets and how to use Beans for managing data.

NO TOKENS
WHAT THE TOUR BUSES DON'T SHOW YOU

Java Time Table
hands-on courses underground

Java Stops	Time		Location
	December	January	
Essential Java for developers		01/31-02/04	Los Angeles
			Boston
			Portland
			England
Intensive Java the runtime environment	OFFERED IN MARCH		Los Angeles
			Boston
			Portland
			England
Essential RMI Java distributed objects			Los Angeles
	12/13-12/17		Boston
			Portland
			England
Transactional Programming with EJB	12/01-12/03		Los Angeles
		01/17-01/21	Boston
			Portland
			England
Essential Web Apps - Java for Java technologies	12/07-12/10		Los Angeles
		01/24-01/28	Boston
			Portland
		01/31-02/04	England

Java is a registered trademark of Sun Microsystems

Training Stations:

Los Angeles
Boston
Portland, OR
England

Registration and Information:

800.699.1932
for the US

08000.562.265
within the UK

+44 1242 525 108
within Europe

DEVELOPMENTOR™
www.develop.com

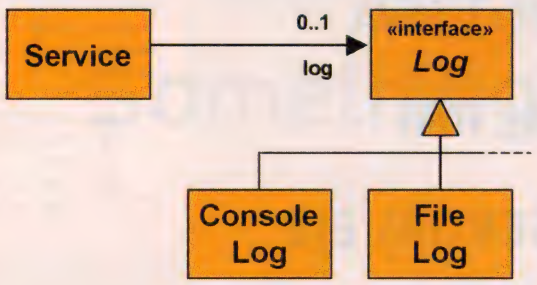


Figure 1. ***UML class diagram of a service with optional support for logging.***

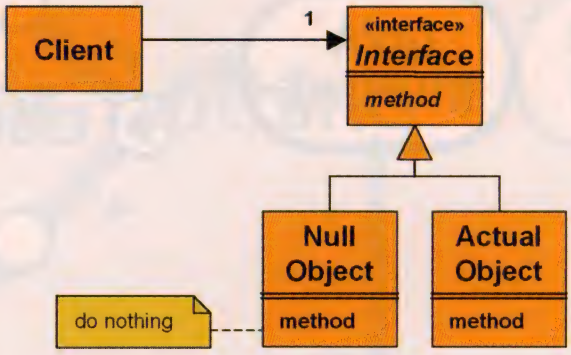


Figure 2. ***Key classes in a* Null Object *collaboration.***

take appropriate inaction, even though the condition is not in any way exceptional, i.e., having a null log is a normal and expected state of the relationship. The condition makes the use of the logging facility less uniform.

A feature of conditional code is that it makes the decision flow of the code explicit. However, this is only a benefit if the decisions taken are important to the logic of the surrounding code; if they are not, then the resulting code is less rather than more direct, and the core algorithm becomes obscured.

Where conditional code does not serve the main purpose of a method, it tends to get in the way of the method's own logic, making the method longer and harder to understand.

LISTING 1.

The root Log interface and sample concrete implementations.

```
public interface Log
{
   void write(String messageToLog);
}

public class ConsoleLog implements Log
{
   public void write(String messageToLog)
   {
      System.out.println(messageToLog);
   }
}

public class FileLog implements Log
{
   public FileLog(String logFileName)
      throws IOException
   {
      out = new FileWriter(logFileName, true);
   }
   public void write(String messageToLog)
   {
      try
      {
         out.write("[" + new Date() + "] " +
                   messageToLog + "\n");
         out.flush();
      }
      catch(IOException ignored)
      {
      }
   }
   private final FileWriter out;
}
```

This is especially true if the code is frequently repeated, as one might expect from the logging facility in the example, or if in-house coding guidelines encourage use of blocks in preference to single statements:

```
if(log != null)
{
   log.write(message);
}
```

The use of an explicit conditional means that the user may choose alternative actions to suit the context. However, if the action is always the same and if the optional relationship is frequently used, as one might expect of logging, it leads to duplication of the condition and its action. Duplicate code is considered to have a "bad smell,"[3] and works against simple changes (fixes or other improvements).

Where the relationship is null, there is no execution cost except a condition test and branch. On the other hand, the use of an explicit test means that there will always be a test. Because this test is repeated in separate pieces of code, it is

LISTING 2.

Initializing and using a Log object in the Service.

```
public class Service
{
   public Service()
   {
      this(null);
   }
   public Service(Log log)
   {
      this.log = log;
      ... // any other initialization
   }
   public Result handle(Request request)
   {
      if(log != null)
         log.write("Request " + request + " received");
      ...
      if(log != null)
         log.write("Request " + request + " handled");
      ...
   }
   ... // any other methods and fields
   private Log log;
}
```

A Can of Worms.

Casting for the best Java jobs?

Cut bait with the online bulletin boards and resume databases. Finding a great Java job is easy with Career Central. We reel in the jobs that match what you want. Best of all, it's free.

Career Central for Java Developers.

Sign up at www.careercentral.com/java

career central®
for java developers

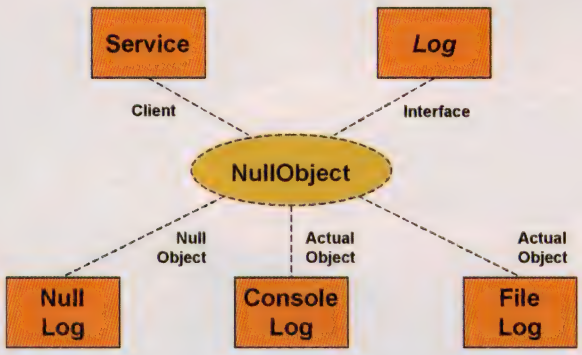


Figure 3. ***The relationship between the service structure and the roles in* Null Object.**

not possible to set debugging breakpoints or print statements that show all uses of a null case.

Solution

Provide a class that conforms to the interface required of the object reference, implementing all of its methods to do nothing or provide return default values. Use an instance of this class when the object reference would otherwise have been null. Figure 2 shows the essential configuration as a class model.

Resolution

Returning to the server example, we can introduce a Null Object class, NullLog, into the Log hierarchy. Such a comfortably null class (see Listing 3) does nothing and does not demand a great deal of coding skill!

The tactical benefit of a Null Object, once initialized, is in simplifying the use of this simple logging framework. It is now guaranteed that the relationship to a Log object is mandatory, i.e., the multiplicity from the Service to the Log is 1 rather than 0..1. This means that the conditional code can be eliminated, which makes the use of logging more uniform and the presence or absence of enabled logging transparent (see Listing 4).

Just as the essential structure for Null Object may be diagrammed in UML, its use in our code can be documented in a collaboration diagram. Figure 3 shows the classes and interfaces in our design and how they correspond, in terms of roles, to the elements outlined in Figure 2.

Consequences

Introducing a Null Object simplifies the client's code by eliminating superfluous and repeated conditionals that are not part of a method's core logic. The relationship moves from being optional to mandatory, and selection and variation are expressed through polymorphism and inheritance rather than procedural condition testing, making the use of the relationship more uniform. However, to preserve this invariant, care must be taken to correctly initialize the reference and to ensure either that it is not changed, i.e., declare it final, or that any change preserves the requirement that null is replaced by Null Object.

A Null Object is encapsulated and cohesive: It does one thing—nothing—and it does it well. This eliminates duplicate coding, makes the (absence of) behavior easier to reuse, and provides a suitable venue for debug breakpoints or print statements.

The absence of side effects in any method call to Null Object means that instances are immutable and therefore shareable and thread-safe, so that a Null Object is a Flyweight.[1] A Null Object is typically stateless, which means that all such instances will have the same behavior, and only one instance is required, suggesting that the object might be implemented as a Singleton.[1]

Using a Null Object in a relationship means that method calls will always be executed and arguments will always be evaluated. For the common case this is not a problem, but there will always be an overhead for method calls whose argument evaluation is complex and expensive.

The use of Null Object does not scale for distribution if passed by reference, e.g., if it implements java.rmi.Remote. Every method call would incur the overhead of a remote call and introduce a potential point of failure, both of which would swamp and undermine the basic *do nothing* behavior. Therefore, if used in a distributed context, either null should be passed in preference to a Null Object reference or pass by value should be used if RMI is the distribution mechanism, i.e., transparent replication rather than transparent sharing becomes the priority. Because a Null Object class is small and simple, class loading is cheap and the only change the programmer needs to make is to implement java.io.Serializable:

```
public class NullLog implements Log, Serializable
{
  ...
}
```

LISTING 3.

A NullLog class providing *do nothing* behavior.

```
public class NullLog implements Log
{
  public void write(String messageToLog)
  {
  }
}
```

LISTING 4.

Introducing a NullLog object into Service.

```
public class Service
{
  public Service()
  {
    this(new NullLog());
  }
  ...
  public Result handle(Request request)
  {
    log.write("Request " + request + " received");
    ...
    log.write("Request " + request + " handled");
    ...
  }
  ...
  private Log log;
}
```

IMAGINE RIDING IN THE FIRST CAR OF A 2,000 MPH ROLLERCOASTER THAT COULD TAKE YOU FROM YOUR DESK TO THE MOON.

Very good. Welcome to the new world of building digital businesses. Not knowing what the future will bring terrifies some people. Others are excited by the possibilities.

If you thrive in this environment we would like to introduce you to Viant.

So What Is This Wild Ride All About?

As you are well aware, the Internet has created an opportunity for businesses around the globe to completely reinvent business. Viant is focused on Building Digital Businesses - helping to create the firms which will shape the future. This is definitely not a ride for everybody.

How can I Get On Board?

As an integrated strategy and solutions partner, Viant has pioneered powerful approaches for successfully incubating and launching e-commerce enterprises for Fortune 500 companies and startups. We're actively seeking Technologists who work collaboratively with Strategists and Creatives to change today's business landscape.

So who are we looking for right now? Well, generally people who love learning and are the best in their fields. More specifically, we are looking for **Technologists, Creatives and Strategists** who want to work collaboratively in multidisciplinary teams. You may want to tunnel deep into Jini™, or develop a broader understanding of the top Java Application Servers, like Weblogic or Websphere. You're smart, fast, and you want to be challenged.

This Is All A Little Frightening. But You Are Not Alone.

Viant boasts one of the most collaborative, passionate, team-based corporate cultures on the planet; we know that our people are our product. For all of your projects you will have the experience of the worldwide Viant organization at your fingertips.

So, Want To Take A Ride?

If so, please go to http://www.viant.com/careers and submit a resume. Sorry, no phone calls, mail or faxes. (See, we really do take digital business seriously!)

VIANT

An Equal Opportunity Employer.

Users of a hierarchy that includes a Null Object will have more classes to take on board. The benefit is that with the exception of the point of creation, explicit knowledge of the new Null Object class is not needed. Where there are many ways of *doing nothing*, more Null Object classes can be introduced. Variations on this theme include the use of an Exceptional Value object,[4] that rather than doing nothing, raises an exception for any method call or returns Exceptional Value objects as a result.

Taking an existing piece of code that does not use Null Object and modifying it to do so may accidentally introduce bugs. By carefully following the Introduce Null Object refactoring,[3] developers can have greater confidence making such changes and avoiding pitfalls. When introducing a Null Object class into an existing system, a suitable base interface may not exist for the Null Object class to implement. Either that part of the system should be refactored to introduce one or the Null Object class must subclass the concrete class referred to by the reference. The latter approach is a little undesirable, as it implies that if there is any representation in the superclass it will be ignored. Such inheritance with cancellation can lead to code that is harder to understand because the subclass does not truly conform to the superclass, i.e., fails the *is a* or *is a kind of* litmus test for good use of inheritance and includes redundant implementation that cannot be uninherited. This approach also cannot be used if the concrete class or any of its methods are declared final. Note that a Null Object class should not be used as a superclass except for other Null Object classes.

Pattern Anatomy

Any pattern is a narrative, essentially a story that unfolds with *dramatis personae*, the to and fro movements of the plot elements, and a happy ending. A pattern is also a guide to construction, telling you when and how to build something, passing on strategies, tactics, and observations from experience. This is as true of software development patterns as it is of building architecture where patterns originated.[5,6]

What You Should Find in a Pattern

A pattern identifies a general approach to solving a problem, typically capturing a solution practice or collaborative structure. It identifies the general context of the problem, the nature of the problem, the interplay of conflicting forces that create the problem and binds its solution, the configuration that resolves the problem, and the resulting context of applying such a solution. In short: where, what, and how the problem arises and is solved. Rather than living in the abstract, a pattern needs to be illustrated by something concrete, such as an example with code and diagrams.

Patterns also tend not to live in isolation, and so a pattern typically contains references to other patterns that complete it. Such links increase the value and usability of a pattern because they lay out some of the possible tracks your development can follow, for instance, the use of Singleton with Null Object. A pattern can also contain discussion of patterns that are in some way similar to it or coincidentally related. However, there is a danger that a pattern written in this style can end up as a comparative pattern study or treatise, i.e., an academic exercise rather than a piece of knowledge focused on practice and problem solving.

Patterns are considered to be about real things, so they are empirical rather than theoretical, and they are about repeated rather than one-time solutions to problems. There is often an onus on pattern authors to demonstrate this, such as by including a listing of some known uses of a pattern that cite, for example, specific frameworks and libraries. We could cite a number of framework examples for Null Object, as it has been discovered time and time again, but as a general pattern—as opposed to a specifically OO design pattern—it is perhaps more common than people realize: Consider the role of the null file device on many operating systems (/dev/null on Unix and NUL on Microsoft systems), no-op machine instructions, terminators on Ethernet cables, etc.

Finally, one of the most significant parts of a pattern is its name. The name allows us to discuss solutions easily and at an appropriate level of abstraction. This suggests that a pattern's name should be drawn from the solution and should be an evocative noun phrase[7] that can be used in ordinary conversation, i.e., to support usage such as "if we use a Null Object..." as opposed to "if we apply the Null Object pattern...." It is not uncommon for a pattern to have different names, either because different people have documented it at different times or because other names have different connotations. For instance, Null Object has also been known as Active Nothing and Smart NIL.

Pattern Form

There are many forms for documenting patterns,[8] ranging from the highly structured heading-based template form[1] to more narrative, literary forms.[5] The essential elements of a pattern can be found in each form, and the choice of form tends to be a matter of personal preference as much as anything else.

A quick scan through pattern sources such as the Patterns Home Page[9] or the *Pattern Languages of Program Design* (PLoPD) books[10–12] should be enough to convince anyone of the diversity of pattern form. Null Object has been documented in a variety of forms, which vary widely in structure and length: from the structured GoF-like form,[2] to the shorter structured form used in this column, to the brief thumbnail-like production rule form.[13]

Conclusion

Patterns can quite literally help create a vocabulary for development, accelerating communication, and the understanding of ideas. They are pieces of literature that capture successful structures and practices, offering developers reusable ideas rather than reusable code. A pattern's form is essentially a vehicle for conveying these.

As a specific example, Null Object illustrates how substituting a stateless object that does nothing offers a surprisingly powerful solution. It can be used tactically in many systems to encapsulate and simplify control flow—something for nothing. ■

Java Business℠
ENTERPRISE SOLUTIONS IN ACTION

JAVA™

DECEMBER 7–9, 1999 | JACOB K. JAVITS CONVENTION CENTER | NEW YORK, NY

Translate Java™ Technology into Competitive Advantage

The importance of information technology in the enterprise has never been greater. To compete and grow in the rapid-paced Internet economy, your organization needs to be responsive—and it is up to you to provide the tools. You need to translate technology in ways that will help your business capitalize on change.

Luckily for you, Java™ technology-enabled solutions not only deliver the strength and productivity your organization needs, but they are nimble as well—making development and deployment both easier and quicker (and every minute counts in the Internet economy).

Learn to accelerate your enterprise with Java technology-enabled solutions at the Java Business℠ conference—the premier source for information on the successful development and deployment of Java technology in business today. Choose from five in-depth tracks delivering up-to-the-minute, real-world information: Build Your Business; Industry in Action; Extend the Enterprise, Technology Direct from the Source, and Java Technology and XML. Experts will show you how to:

- Cut costs and shorten development cycles.
- Benefit from greater portability and flexibility with high-quality, reusable code.
- Support multiple distributed computing models.
- Preserve and extend legacy systems.
- Integrate disparate client/servers and enable distributed solutions.
- Deploy applications that are scalable, extensible and maintainable.
- Leverage the freedom of Jini™ technology-enabled media and devices.

Java technology is the workhorse, powering the enterprise while saving time and money. Making you the hero, today and tomorrow.

Register today or get conference information at www.javabusinessconference.com or call 888-886-8309 (International 781-433-1511). **Use priority code MTZZMTH.**

©1999 ZD Events Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Java, the Java Coffee Cup logo, Java Business, the Java Business logo, Jini and all Java-based marks and logos are trademarks or registered marks of Sun Microsystems, Inc. in the United States and other countries.

Sun microsystems

Go to www.javabusinessconference.com

ZD ZIFF-DAVIS a SOFTBANK company

Modeling and Java / *Craig Larman*

Continued from page 68

ware representation." Nope. The CEO will look at you sideways and ask, "What the hell are you talking about?" Consequently, we have to give politically astute answers such as "because it will increase software reuse" even though there is precious little of that going on, except in common libraries and frameworks. Of course, object technology is also of value because it can support the design of elegant, loosely coupled systems that scale and extend more gracefully (and less costly) than the alternatives, which is another key reason for its adoption.

Conclusion

Now that we have chosen a class of object in the domain layer to receive a system request (e.g., RentItemsHandler receives messages from the presentation layer and delegates them), it is time to focus on the design of the domain layer in which objects fulfill application logic responsibilities. ■

References

1. Larman, C., "Transition from analysis to design," Java Report, Vol. 4, No. 9, Sept. 1999, pp. 62–68.
2. Larman, C., "The Use Case model—what are the processes?" Java Report, Vol. 3, No. 8, Aug. 1998, pp. 62–72.
3. Larman, C., Applying UML and Patterns—An Introduction to Object-Oriented Analysis and Design, Prentice Hall, 1997.
4. Gamma, E., et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison–Wesley, 1995.

JavaScripting / *Steven W. Disbrow*

Continued from page 72

LISTING 2.

thanks.htm.

```
<HTML>
<HEAD>
<TITLE>Thanks!</TITLE>
</HEAD>
<BODY>
Thanks for the info!
</BODY>
</HTML>
```

browsers support <DIV>, but <LAYER> is a Netscape-only tag.) However, try as I might, I've never been able to get Netscape to respect the "visibility" attribute of a <DIV> tag. So, I've resorted to this combination of tags to create the same effect.

You might be thinking that the "layers" approach would also work for IE. Indeed it would. However, for the purpose of this article, I've included the "delay and submit" technique just to show how it works.

Nothing Is Perfect

While this technique works quite well for IE, Netscape's insistence on contacting the mail server makes it a less-than-perfect solution for clients using that browser. Even in an "always connected" environment, there will be occasional mail server outages, during which any messages sent via this technique will be lost. However, if you can't use or don't have access to server-side email helper utilities, the techniques shown here will allow you to easily send nicely formatted emails from any of your Web pages. ■

give me liberty
GIVE ME
thebazaar
The Technology of Cooperation
Yearning to see open source software get full recognition in the real world of enterprise computing? The Bazaar welcomes IT professionals who want information and training on how to integrate open source software into their existing systems. Join us and the master minds of open source technology for an in-depth exploration in over 75 conference tracks and hands-on training sessions. — ALL STUDENTS IN OUR TUTORIALS GET THEIR OWN PC!
Give us your systems analysts, your CIO's, your software developers...
Featured Speakers include:
Michael Cowpland, CEO, Corel Corporation
Jon "Maddog" Hall, President, Linux International
Ralph Nader, Consumer Advocate
Richard Stallman, Founder, Free Software Foundation
Bob Young, CEO, RedHat Incorporated
PRESENTED BY
EARTHWEB
JACOB K. JAVITS CENTER, NEW YORK, NY DECEMBER 14-16, 1999
Register today at www.thebazaar.org
©1999 EarthWeb Inc. All rights reserved. EarthWeb and the EarthWeb logo are registered trademarks of EarthWeb Inc.

ADVERTISER INDEX

Advertiser	Page	URL	e-mail
Ad Astra Engineering	45	www.JumpingBeans.com	crygaard@AdAstraEng.com
Applied Reasoning Systems	33	www.appliedreasoning.com	jmartin@appliedreasoning.com
BEA	2	www.weblogic.beasys.com	info@weblogic.com
C++ World	41	www.cwconference.com	conferences@sigs.com
Career Central	77	www.careercentral.com/java	javainfo@careercentral.com
Cyrus Intersoft	13	www.cyrusintersoft.com	leslie@cyrusintersoft.com
DevelopMentor	75	www.develop.com	
EON Technologies, Inc.	29	www.bankframe.com	info@bankframe.com
Fiorano Software, Inc.	86	www.fiorano.com	info@fiorano.com
Flashline.com	31	www.flashline.com	info@flashline.com
Force 5	47	www.force5.com	steve@force5.com
Gemstone Systems	37	www.gemstone.com/guts	info@gemstone.com
Geek Cruises	52	www.geekcruises.com	neil@geekcruises.com
IAM Consulting	71	www.iam-training.com	java@iam-training.com
IBM	9	www.ibm.com/software/soul/db2	
IBM	25	www.ibm.com/developerWorks	devWorks@us.ibm.com
Insignia Solutions	1	www.insignia.com	evm@insignia.com
Intel	27	http://developer.intel.com/vtune	mpl@mailbox.intel.com
Intuitive Systems	61	www.optimizeit.com	optimizeit@intuisys.co
InstallShield	C3	www.installshield.com	sales@intallshield.com
Java Business Conference	81	www.javabusinessconference.com	
JHL Computer Consultants	50	www.jhlcomp.com	jhlcomp@gate.net
KL Group	C4, 48, 53	www.klg.com	info@klgroup
Loox	59	www.loox.com	info@loox.com
NuMega	65	www.numega.com	julie.corvo@numega.com
Object Design	51	www.objectdesign.com/javlin	info@objectdesign.com
Object Insight	73	www.object-insight.com	info@object-insight.com
Object People	55,87	www.objectpeople.com	rausten@objectpeople.com
OOP 2000	63	www.oopconference.com	infogmbh@sigs.com
Oracle	18-19	www.oracle.com/mobile/	
Parasoft	35	www.parasoft.com	Janusz@parasoft.com
Persistence Software	5	www.persistence.com	info@persistence.com
PreEmptive	69	www.preemptive.com	info@preemptive.com
RSA Data Security	C2	www.rsa.com/win	lstrand@rsa.com
SilverStream	23	www.silverstream.com	info@silverstream.com
SoftWired	62	www.softwired-inc.com	softwired@sofwired-inc.com
Sybase	39	www.sybase.com/easerversuccess	sales@sybase.com
Symantec	21	www.visualcafe.com	
The Theory Center	17	www.theorycenter.com	info@theorycenter.com
ThinWEB.com Corp.	15	www.thinweb.com	mweber@thinweb.com
Tidestone Technologies	43	www.tidestone.com	sales@tidestone.com
TogetherSoft	57	www.togethersoft.com	coad@togethersoft.com
Tower Technology	67	www.towerj.com/jr	sales@twr.com
Viant Corporation	79	www.viant.com	cbryson@viant.corp
Zero G Software	7	www.zerog.com	info@zerog.com

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

BPA INTERNATIONAL

ADVERTISING/SALES

Senior Sales Director/Western Region Manager
Miles E. Silverman
• phone 212.242.7447 • fax 212.242.7574
• email msilverman@sigs.com

Advertising Sales Manager, Eastern Region
Brian Levine
• phone 212.242.7447 • fax 212.242.7574
• email blevine@sigs.com

Advertising Sales Manager
Development Solutions Buyer's Guide
Maurine Blake
• phone 212.242.7447 • fax 212.242.7574
• email mblake@sigs.com

CIRCULATION

Vice President Circulation
Christine Oldenbrook

Circulation Manager
Kimberly Chilcutt

FINANCE

Vice President Finance
Margherita R. Monck

Controller
David Chatterpaul

Accounts Receivable Credit Manager
Rey Shields

Accounts Payable
Bibi Budhram

MANUFACTURING

Vice President, Manufacturing
Brad Stauffer

Sr. Advertising Production Manager
Jennifer A. Burrow

Advertising Production Coordinator
Joleen Mathews

FOUNDER
Richard P. Friedman

101 communications

SIGS PUBLICATIONS

WORLDWIDE OFFICES

Jane Major, SIGS Ltd., Brocus House,
Parkgate Road, Newdigate, Surrey RH5 5AH, UK
• phone 011.44.1.306.631.331 • fax 011.44.1.306.631.696
• email enquiries@sigs.com

Sabine Schiffman, SIGS Conferences GmbH, Odenthaler Str. 19, D-51465 Bergisch Gladbach, Germany
• phone 49.2202.9372.0 • fax 49.2202.9372.2
• email 100634,2070@compuserve.com

Java Report 1999 Author Index

A

Ackerley, John, Alex Koutsoumbos, Murtuza Hashim, and Angelo Maffione
"Java Server Pages and Servlet Design," 4(12) Dec. 1999, 24–30.

Allin, Jonathan
"Chaos and BigDecimals," 4(1) Jan. 1999, 39–48.

Altherr, Marcel, Martin Erzberger, and Silvano Mattheis
"SoftWired iBus—Middleware for the Java Platform," 4(12) Dec. 1999, 54–62.

Añez, Juancarlo ***Product Review***
"Java XML Parsers," 4(2) Feb. 1999, 17–26.
"Analyze Your Code With Compuware's NuMega DevPartner for Java" 4(8) Aug. 1999, 12–17.
"Achieving Synergy, ObjectSpace's Voyager ORB 3.0," 4(12) Dec. 1999, 14–22.

B

Ball, Steve ***Effective Java***
"Supercharged Strings," 4(2) Feb. 1999, 62–69.

Bosch, Peter
"Effective Multithreading," 4(1) Jan. 1999, 31–38.

Bragg, Tom ***Product Review***
"Parasoft's CodeWizard for Java," 4(3) Mar. 1999, 12–17.

Brown, Kyle and Gary Craig
"Using Java Server Pages—Servlets Made Simple," 4(8) Aug. 1999, 41–50.

Brown, Kyle, Gary Craig, Greg Hester, and Peter Jakab
"Using Server-Side Java Successfully," 4(6) Jun. 1999, 51–60.

C

Crawford, John Miller ***Effective Java***
"Native territory," 4(3) Mar. 1999, 65–71.
"Interfaces—the outside story," 4(7) Jul. 1999, 58–66.

Crupi, John ***The Architect's Corner***
"Designs that capitalize on the latest Java™ technologies," 4(9) Sept. 1999, 72, 69–70.

D

Davis, Mark ***Durable APIs***
"Designing durable APIs in Java," 4(2) Feb. 1999, 74–77.
"Immutables," 4(4) Apr. 1999, 70–77.
"Abstraction," 4(6) Jun. 1999, 80–89.
"Serialization," 4(8) Aug. 1999, 65–75, 84.

Davis, Mark and Helena Shih
"The Java International API: Beyond JDK 1.1," 4(2) Feb. 1999, 27–36.

Deadman, Richard
"XML as a Distributed Application Protocol: The Technology and the Politics," 4(10) Oct. 1999, 16–21.

Deugo, Dwight ***Editor's Note***
"The right to choose," 4(1) Jan. 1999, 4.
"Looking for JDK 1.2?" 4(2) Feb. 1999, 6–8.
"The cost of Java," 4(3) Mar. 1999, 6, 22.
"Will the increase ever stop?" 4(4) Apr. 1999, 6.
"'Ready-to-apply' expertise," 4(5) May 1999, 8.
"JavaOne," 4(6) Jun. 1999, 8.
"Moving Java to where it's needed," 4(7) Jul. 1999, 6.
"The revolution," 4(8) Aug. 1999, 6.
"Ideas," 4(9) Sept. 1999, 6.
"So, What's XML?," 4(10) Oct. 1999, 6.
"Taking care of business," 4(11) Nov. 1999, 6.
"The best things in life are free," 4(12) Dec. 1999, 6.

Disbrow, Steven ***JavaScripting***
"Client-Side processing?" 4(2) Feb. 1999, 70–73.
"Your own private Idaho(),," 4(4) Apr. 1999, 66–69.
"Linking JavaScript objects with HTML controls," 4(7) Jul. 1999, 48–54.
"Enhancing HTML controls with JavaScript objects," 4(9) Sept. 1999, 58–60.
"Building better email with JavaScript," 4(12) Dec. 1999, 70–72, 82.

Domajnko, Tomaz, Matjaz B. Juric, Bostjan Brumen, and Ivan Rozman
"How to Store Java Objects?" 4(4) Apr. 1999, 33–46.

F

Fisher, Mark
"JavaHelp for the Technical Writer," 4(4) Apr. 1999, 47–54.
"JavaHelp for the Java Programmer," 4(6) Jun. 1999, 21–32.

Frankel, David S. ***Business Objects***
"The OMG Meta-Object Facility," 4(3) Mar. 1999, 56–62, 71.
"The UML Meta-Model," 4(6) Jun. 1999, 90–102.
"CORBA components—alive and well," 4(10) Oct. 1999, 70–77.

Freedman, Gordie
"The Right Tools for the Job—Common Performance Issues & Solutions," 4(7) Jul. 1999, 29–34.

Fregly, Christopher A.
"Flexible Strategies for Optimizing Application Performance," 4(10) Oct. 1999, 48–58.

Frascadore, Greg
"Java Application Server Security Using Capabilities," 4(3) Mar. 1999, 31–42.

G

Gamma, Erich and Kent Beck
"JUnit: A Cook's Tour," 4(5) May 1999, 27–38.

Gill, Philip J. ***Real World Java***
"Arch Gets a Dose of Reality," 4(1) Jan. 1999, 18–22, 66.
"Online Mortgage Company Invests in Java, CORBA Combination," 4(3) Mar. 1999, 18–22.
"Java Perks Up E-Commerce Site," 4(6) Jun. 1999, 18–20.
"Trading in Real Time With Java, Instinet" 4(10) Oct. 1999, 12–14.

H

Haggar, Peter
"Effective Exception Handling in Java," 4(4) Apr. 1999, 55–65, 77.

Hardy, Vincent ***2D Graphics***
"Understanding Java 2D™: Shapes," 4(1) Jan. 1999, 70–79.
"Understanding Java 2D™: AffineTransforms,"4(5) May 1999, 68–76.
"How to use AffineTransforms to create animation effects," 4(8) Aug. 1999, 76–80.

Hay, Phillip
"The Bumpy Road Using Java to Implement a User Interface," 4(11) Nov. 1999, 57–62.

Henney, Kevlin ***Patterns in Java***
"Something for nothing," 4(12) Dec. 1999, 74–80.

Houlding, David
"Delivering Web Services Using CORBA," 4(9) Sept. 1999, 17–28.

Hunt, John
"The Collection API," 4(4) Apr. 1999, 17–32.

J

Jewell, Tyler
"Greater Garbage Collector Access with Reference Objects," 4(6) Jun. 1999, 69–78.

Jones, Steven
"Error Handling with XML," 4(10) Oct. 1999, 22–32.

Jordan, David ***ODMG***
"Serialization is not a database substitute," 4(7) Jul. 1999, 68–76, 79.
"Database trends," 4(11) Nov. 1999, 64–73.

K

Kaplan, Lowell ***Java Means Business***
"Application servers: distributed development and deployment made easy," 4(1) Jan. 1999, 67–68, 79.
"Distributed computing in Java: RMI, CORBA, or both?," 4(7) Jul. 1999, 80, 78–79.

Kumaran, S. Ilango
"A Quick Look At Jini," 4(8) Aug. 1999, 51–54.

L

Landis, Sean
"Distributed Event Notification Using RMI," 4(7) Jul. 1999, 17–28.

Larman, Craig ***Modeling and Java***
"The System Behavior Model: What it does," 4(3) Mar. 1999, 72–78.
"What the UML is—and isn't," 4(5) May 1999, 20–24.
"Transition from analysis to design," 4(9) Sept. 1999, 62–68.
"The Presentation and Domain Layers," 4(12) Dec. 1999, 64–68, 82.

Lauinger, Todd
"Object-Oriented Software Development in Java," 4(2) Feb. 1999, 59–61, 73.

Lipton, Daniel
"A Reusable Email Service: Corba Meets JavaMail," 4(6) Jun. 1999, 61–68.

Long, Peter
"To Do Or Not To Do—That Is the Question," 4(5) May 1999, 39–50, 66.

M

Mak, Ronald
"How to Create Bipolar CORBA Objects in Java," 4(9) Sept. 1999, 29–37.

Malarvannan, Mani
"Applying Reflective Architectural Pattern to EJB," 4(8) Aug. 1999, 31–40.

Malks, Dan ***The Architect's Corner***
"JSP and Servlets: A powerful pair," 4(11) Nov. 1999, 80, 74–78.

Mannion, Mike
"Concurrent Contracts: Design-By-Contract™ and Concurrency in Java," 4(5) May 1999, 51–66.

Matic, Nebojsa
"Email Framework," 4(11) Nov. 1999, 41–48.

McManus, Alex
"Layout Managers—Love Them Or Leave Them, They're Here to Stay!" 4(6) Jun. 1999, 33–50.

Metsker, Steven J.
"Java Permutations and Combinations," 4(1) Jan. 1999, 49–60.

Metz, Eldon
"Advanced JavaBeans Development," 4(12) Dec. 1999, 32–42.

Mohseni, Piroz
"XML: The Data Spice for Java," 4(11) Nov. 1999, 49–54.

Monson-Haefel, Richard
"What's New in EJB 1.1," 4(8) Aug. 1999, 23–30, 83.

N

Nylund, Joel
"Memory Leaks in Java Programs," 4(11) Nov. 1999, 22–30.

O

O'Shea, John
"Locating CORBA Objects From Java," 4(3) Mar. 1999, 23–30.

Oaks, Scott ***Scott's Solutions***
"Printing in Java," 4(2) Feb. 1999, 80, 78–79.
"Versioning solutions," 4(4) Apr. 1999, 80, 78–79.
"Using RMI callbacks to deal with asynchronous messages," 4(6) Jun. 1999, 112, 109–110.
"Dealing with I/O in an event-oriented fashion," 4(8) Aug. 1999, 88, 86–87.
"Using the Reflection API," 4(10) Oct. 1999, 80, 78–79.
"Creating good Java interfaces," 4(12) Dec. 1999, pp. 88, 86.

P

Patten, Bob and Gary Grandlienard
"Using Resource Bundles to Internationalize Text," 4(2) Feb. 1999, 37–44.

Patten, Bob and James McCabe
"Reducing Network Traffic in Distributed Java Apps," 4(9) Sept. 1999, 51–57.

R

Rems, Tony ***Product Review***
"BEA System's WebLogic Application Server," 4(9) Sept. 1999, 12–16.

Repking, Ron and Christopher Nippert
"Developing Successful Enterprise Applications Using Java Servlets," 4(9) Sept. 1999, 39–50.

Riehle, Dirk and Erica Dubach
"Working With Java Interfaces and Classes, How to Separate Interfaces From Implementations," 4(7) Jul. 1999, 35–46.
"Working With Java Interfaces and Classes: How to maximize design and code reuse in the face of inheritance," 4(10) Oct. 1999, 34–46.

S

Schecter, Robb
"Object-Oriented Widgets," 4(12) Dec. 1999, 44–52.

Schedlbauer, Martin and David Udin
"How to Successfully Migrate to Java," 4(1) Jan. 1999, 61–66.

Seshadri, Govind ***Distributed Computing***
"Implementing custom sockets with Java/RMI," 4(5) May 1999, 77–85.
"Remote object activation," 4(10) Oct. 1999, 60–68.

Shepherd, Pat ***Product Review***
"Java Studio: Dynamic Demonstrations of Java and Component-Based Development," 4(1) Jan. 1999, 10–17.

Snyder, Scott
"JavaServer Pages for Enterprise-Level Computing," 4(3) Mar. 1999, 43–55.

T

Tillinghast, Tig ***Madison Avenue Java***
"How the Web is killing Java," 4(1) Jan. 1999, 80.
"Fuzzy logic and Java-enabled beard trimmers," 4(3) Mar. 1999, 80.
"The ongoing fight to define Java's identity," 4(5) May 1999, 88, 86.

V

Viswanathan, K.V.
"Exploiting Java's Object-Oriented Features," 4(8), Aug. 1999, 57–64, 83.

W

Waters, John K. ***Coffee Talk***
"The Glue That Holds Diverse Systems Together: An Interview with Symantec's Mansour Safai" 4(2) Feb. 1999, 14–16, 26.
Special Feature
"The Hottest and Coolest Products for '99!" 4(11) Nov. 1999, 11–20.

Weeks, Kevin ***Product Overview***
"Java Education 101," 4(5) May 1999, 16–18.

Weisfeld, Matt and Gabriel Torok
"Fine Tuning Java Applications," 4(11) Nov. 1999, 32–40.

Werner, Laura
"Efficient Text Searching in Java," 4(2) Feb. 1999, 45–58, 77.

Wirfs-Brock, Allen
"Complex Java Applications—Breaking the Speed Limit," 4(1) Jan. 1999, 23–30.

Z

Zino, Todd C. ***Product Reviews***
"Class Libraries That Integrate Into the Existing AWT: JViews encourages 2D developers to take Java plunge," 4(4) Apr. 1999, 14–16.
"JProbe 2.0 Developer Suite Offers Sophisticated Performance Utilities," 4(7) Jul. 1999, 12–16.

Continued from page 88

not for the communication between client and server. But we still might want to use an interface for the class that does all the socket communication.

There are other permutations that we could list, but the question that interests us is whether we could write a general interface such that it doesn't matter whether or not we implement the code with RMI or CORBA or our own network programming.

The key to this is to attempt to limit the number of dependencies that we have in the interface while at the same time having all the functionality that we need. That means in particular that we cannot afford to use an interface that extends another interface because that influences the eventual implementation.

But the similarity all these network-based implementations will share is that the actual call may be subject to network failure. RMI points this out explicitly in its interfaces by requiring all methods to throw a remote exception. CORBA interfaces do not show this explicitly, but CORBA calls that fail will throw a runtime exception, typically, a subclass of SystemException, which is the superclass in the CORBA package for all CORBA exceptions. And your own socket-based class will throw socket exceptions, or I/O exceptions, or whatever.

So it makes sense for your interface to indicate the possibility of a communications-based failure. This is often done by using the RemoteException class as a catch-all class for communication exceptions; that's a particularly good class to use because it can embed another exception within it so that the programmer can figure out what the actual exception is. And even though the RemoteException class is in the java.rmi package, it doesn't really introduce any dependencies on RMI—that's just where it lies in Java's class hierarchy.

A generic interface, then, might look like this:

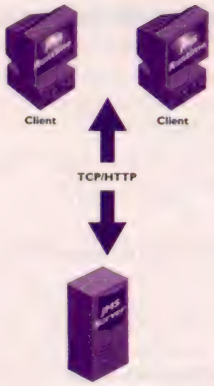
```
public interface Adder {
    public int add(int i, int j) throwsRemoteException;
}
```

Now we can implement our server with RMI by creating a new interface that extends Adder and Remote; when the RMI stub that implements that interface is passed to the client, the client will just use it as an Adder. We can similarly implement our server with CORBA by providing the interface that extends Adder, Object (in the CORBA package), and IDLEntity. And we can create a new class that implements Adder, uses sockets to talk back and forth between client and server, and wraps all its socket and IOExceptions into RemoteExceptions.

The point, of course, is that Java is a distributed programming language, and it takes some forethought to design good interfaces that fit well into its distributed paradigm. If you ever think that you may need to use a generic interface in a distributed manner, you should go ahead and put in the remote exception possibilities. If you think that you might someday reimplement your RMI server in CORBA, you might use a generic interface (with remote exceptions) in the client so that it is protected from such changes. ■

JMS
High Performance
Fiorano 3.0
Enterprise Messaging Server
Secure
SCALABLE

Build secure, reliable and highly available distributed systems using standards-based Java APIs.



Communication, Security, Administration

- 100% Java implementation of **JMS** (Java Message Service) publish/subscribe and point-to-point communication APIs.
- Available C++ runtime library allows C++ and Java programs to communicate seamlessly.
- Lightweight, efficient and robust environment; thin runtime access to messaging services.
- ACL-based security based on Java 2 standards allows external control over system resources. Integrated SSL support for encrypted data transfers.
- Comprehensive Administration API for remote monitoring of servers.
- Integrates with popular Application Servers.

Guaranteed Delivery, Scalability and Availability

- Automatic store-and-forward ensures guaranteed message delivery across faulty networks.
- Server clustering allows support for thousands of concurrent client connections. Built-in support for load balancing, failover protection and high-availability allows clients to automatically connect to the least loaded server in a cluster.

Fiorano

Fiorano Software, Inc.
236 N. Santa Cruz Ave., Suite #213 • Los Gatos, CA 95030
Phone: (408) 354-3210 • Fax: (408) 354-0846
email: info@fiorano.com • www.fiorano.com

TOPLink does more.

TOPLink speeds application development

TOPLink is a product line of advanced object/relational mapping tools that allow you to map complex business objects and EJBs to relational databases so you can leverage your investment in data *and* deploy new applications with Java.

With TOPLink, you can save as much as 40 percent of the time you would otherwise spend getting your Java application up and running.

TOPLink incorporates a meta-data architecture that separates the details of persistence from the business logic. No SQL programming is required. TOPLink supports any JDBC-compliant database including DB2, Oracle, Sybase, Informix and SQL Server (also new JDBC 2.0 support), as well as non-relational data sources.

For a **FREE** 30-day evaluation copy of TOPLink, or more information, contact us by e-mail at info@objectpeople.com, phone (613) 225-8812, or visit our Web site.

TOPLink for Java 2.0 comes with these great features:

- Visual Builder tool for creating mappings
- Flexibility with 15 types of mappings
- Object-level transactions
- Queries in terms of objects, not SQL
- Transparent 3-tier support to provide remote session and unit-of-work functionality
- Object caching and identity
- Java 2 support for collections and weak referenced caches
- Optimistic and pessimistic locking
- Batch reading and writing
- Support for Oracle8*i*
- Support for **Enterprise JavaBean** persistence, including:
  - Bean-managed persistence
  - Transparent mapping of EJBs to relational databases
  - Fully integrated functionality with Session beans
  - JTS integration

TOPLink for BEA WebLogic gives you application server support by building on the existing EJB support and mature feature set of TOPLink for Java, and providing fully integrated object/relational mapping for container-managed Entity bean persistence with the BEA WebLogic application server.

Contact us about other application server support.

TOPLink™
THE OBJECT PEOPLE

100% PURE JAVA

www.objectpeople.com

CANADA • U.S.A. • U.K. • GERMANY • TURKEY

TOPLink is a trademark of The Object People, Inc. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Other brand and product names are trademarks or registered trademarks of their respective owners.

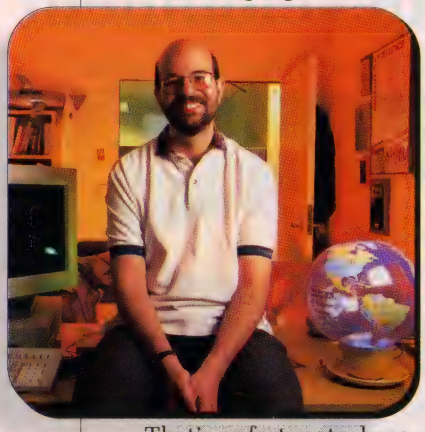
THE OBJECT PEOPLE
www.objectpeople.com

Scott Oaks / soaks@sigs.com

Creating good Java interfaces

WE'RE GOING TO do something a little different this time: we're going to answer the question of what makes a good interface in Java. It's a different sort of question because there is no final answer to it, and much of what I have to say is based on my own opinion. There's some rationale here, but none of the "it works this way" sort of answers I usually try to give.

It's a critical question, especially when we're interested in writing good object-oriented code that can be easily shared among developers—whether they're colleagues down the hall, coworkers in different states, or people who we have never even met before.



The importance of this question stems from the fact that one of Java's valuable assets is the easy way in which we can separate interface from implementation. For programmers like me who were trained in procedural methods, the power behind interfaces is sometimes under-appreciated, and interfaces are therefore under-utilized. That's unfortunate, because sharing a well-designed interface is a simple way to increase the productivity of teams of programmers.

So what things do you need to put into your interface? Let's say that you are writing a class that adds two numbers; we might start with this interface:

```
public interface Adder {
    public int add(int i, int j);
}
```

That's a good interface, but what if we wanted to add doubles? Or numbers represented by our ComplexNumber class? Should we have additional methods for all of these cases or should we have a single method that requires two objects as parameters and returns an object (similar to the way the equals() method works)?

Scott Oaks is a Systems Engineer for Sun Microsystems, where he focuses on practical applications of Java technology. He is the co-author, with Henry Wong, of *Java Threads*.

That's a hard question to answer (and I'll note that the question arises not only in interface design but in class design). It's tedious to have to wrap primitive integers into Integer objects and there's a performance penalty associated with doing so (not to mention all the work it creates for the garbage collector). On the other hand, we could never hope to come up with all possibilities of types that we might eventually want to add—and if we add a method that accepts complex numbers as arguments, then our simple interface becomes dependent on the ComplexNumber interface (or class).

So my general rule of thumb for these cases is to put in the most common argument lists for convenience, and also to put in a method that operates solely on Object variables as a way to handle all the other cases. This gives us a degree of ease of programming, it doesn't make our interface dependent on another interface, and it provides an interface capable of handling even the most general case.

Now, what happens when Acme Computer Company comes out with a really fast computer. How fast? Fast enough that it can accept a network connection, read input from the connection, add two numbers, and write the result back in less time than your Pentium III 500-MHz desktop can invoke a method. So now we want to use the Adder interface to interface to a remote server.

If we use RMI to get to the remote server, we'd typically write the interface as follows:

```
public interface Adder extends Remote {
    public int add(int i, int j) throws RemoteException;
    ...
}
```

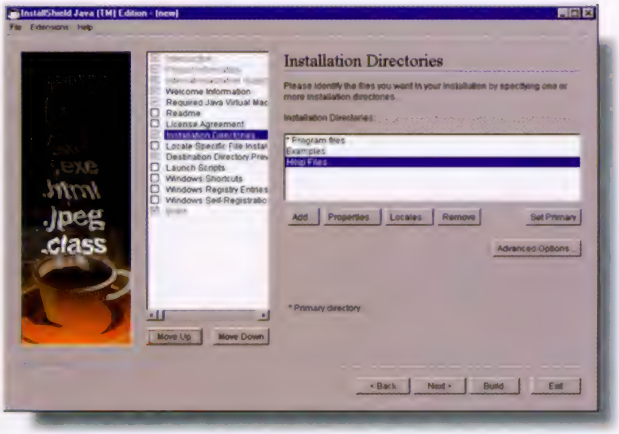
This introduces some dependencies on our eventual implementation: the Remote interface happens to be empty, so that isn't a big deal, and the methods may throw a remote exception.

If we used CORBA to get to the remote server, we wouldn't write an interface at all; we'd write IDL, and the IDL compiler would produce our Java interface. The interface would extend the Corba Object and IDLEntity interfaces, thus introducing those dependencies on the eventual implementation.

If we wanted to write a socket-based server, then we probably wouldn't use an interface at all, at least

Continued on page 86

InstallShield® Java™ Edition Version 2.5

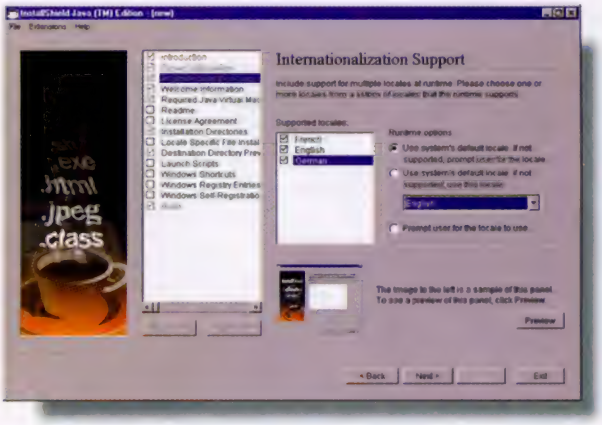


Multiple source and target directory support.

Through the InstallShield Java Builder, you can deploy files to multiple directories on a target system. As you build your setup, locate files in a clear, customized set of directories that will remain correctly distributed on your customer's machine after installation. No more cluttered directories!

Internationalization support.

You can display your custom extensions in different languages. InstallShield Java Edition now includes language resources for all standard end-user dialogs in French, German, Italian, Japanese, and Spanish. Use the **new internationalization APIs** to update your custom extensions, and your setup is ready for the world!



French German Italian Japanese Spanish

NEW VERSION!

InstallShield® Software Corporation

The total solution for Java™ deployment.

InstallShield® Java Edition provides developers with a visual tool designed to create a single Java installation that's consistent in look, feel, and functionality across multiple platforms. New InstallShield Extensions for the Java platform bring the power of Java to installation building, allowing you to plug your own Java-based extensions into your installation. And the InstallShield Installation Applet lets you create a seamless cross-platform launch site for your installation. Best of all, InstallShield Java Edition is authored by InstallShield Software Corporation, the worldwide leader in installation and distribution technology.

Exclusive features in InstallShield Java Edition 2.5

- True cross-platform installations—no need for multiple installations for individual platforms
- InstallShield Extensions for the Java platform give you the power of Java to build your installations—plug in your own Java-based extensions or use one of the pre-built examples
- InstallShield Installation Applet lets your users specify or get the appropriate Virtual Machine (VM), download, and install your Java application by clicking a Web link
- Self-extracting installations allow users to launch installation with point-and-click ease
- Command-line access allows you to add installations to your batch file building process
- Specify multiple source and multiple target directories for your installations
- Create installations in English, French, German, Italian, Japanese, and Spanish

©1999 InstallShield Software Corporation. All rights reserved. InstallShield is a registered trademark and service mark of InstallShield Corporation. "It All Starts Here" is a service mark and a registered trademark of InstallShield Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and/or other countries. Part No. 181-20009 0199

It All Starts Here®

900 National Parkway, Suite 125
Schaumburg, IL 60173-5108 USA
Sales: +1-847-240-0618
Fax: +1-847-619-0788
Toll-free: +1-800-440-8149
sales@installshield.com
www.installshield.com

InstallShield SOFTWARE CORPORATION

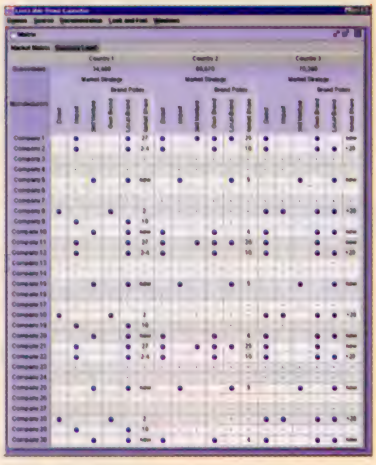
www.installshield.com/java

Real Power for Enterprise Java apps!

JCLASS LiveTable *the enterprise ready grid control*

JClass LiveTable is a scalable, high performance Java grid that lets you quickly build dynamic tables and forms for your Java GUIs. JClass LiveTable gives you complete control over the look and feel of your application. What's more, automatic data-binding with virtually any data source makes JClass LiveTable the ideal solution for enterprise applications. Add to that support for real-time user interaction and easy integration with other JClass components, and it's no wonder JClass LiveTable is the world's first choice in Java grids.

- Real-time scrolling with smooth repainting
- Supports virtually any amount of data (up to 2 billion x 2 billion cells)
- Gives you total control over fonts, color and images
- Supports pluggable look and feel
- Versatile data access and real-time data updating
- Use as a data-entry form, multicolumn list or data grid
- Available for JDK 1.1 and Java 2

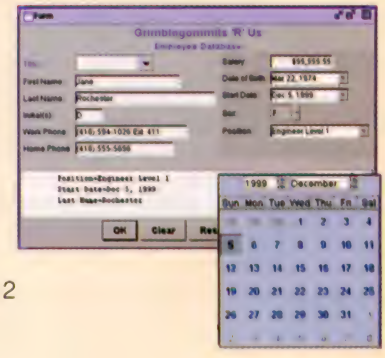


www.klgroup.com/livetable

JCLASS Field *bullet-proof data entry and validation controls*

Never waste time writing complex validation code again! JClass Field is the ultimate collection of data entry and validation components for your Java applications. Multiple field input types, edit and display masks give you the flexibility you need for even the most complex applications. What's more, JClass Field integrates seamlessly with JClass LiveTable, adding up to real power for your enterprise apps!

- Validate strings, floating point values, time or date values, and integer values
- Full support for locale-specific date and time, pick-lists, spin- and combo-boxes
- Includes a general purpose pop-up calendar
- Available for JDK 1.1 and Java 2



www.klgroup.com/field

JCLASS

KL GROUP
Software Development Productivity

KL Group Inc. Toll-Free: 1-888-361-3264 Tel: 416-594-1026 Fax: 416-594-1919
KL Group Europe Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26
www.klgroup.com

© Copyright 1999 by KL Group Inc. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in US and other countries. JClass, JClass Field and JClass LiveTable are trademarks of KL Group Inc. All other products are trademarks or registered trademarks of their respective companies. JR002

expertise is everything

subscribe on-line at www.javareport.com

Yes! I want Java Report for only **$19.50**. That's half-off the regular subscription rate!

☐ bill me ☐ payment enclosed

name

title

company

address

city state zip

phone fax

e-mail address

By providing your e-mail address, you are granting Java Report permission to contact you regarding your subscription, as well as other products and services of SIGS/101communications and/or outside parties. Please indicate those you do NOT wish to contact you.

☐ Java Report ☐ Other SIGS/101communications products ☐ Outside parties

New subscriber Discount! Save 50% if you act now!

Java™ REPORT

Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in US dollars plus shipping. Canada & Mexico $64/year, all other countries $88/year. Canadian GST included.

B91210

Yes! I want Java Report for only **$19.50**. That's half-off the regular subscription rate!

☐ bill me ☐ payment enclosed

name

title

company

address

city state zip

phone fax

e-mail address

By providing your e-mail address, you are granting Java Report permission to contact you regarding your subscription, as well as other products and services of SIGS/101communications and/or outside parties. Please indicate those you do NOT wish to contact you.

☐ Java Report ☐ Other SIGS/101communications products ☐ Outside parties

New subscriber Discount! Save 50% if you act now!

Java™ REPORT

Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in US dollars plus shipping. Canada & Mexico $64/year, all other countries $88/year. Canadian GST included.

B91210

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 2 BRENTWOOD TN

POSTAGE WILL BE PAID BY ADDRESSEE

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

PO BOX 5050
BRENTWOOD TN 37024-9713

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 2 BRENTWOOD TN

POSTAGE WILL BE PAID BY ADDRESSEE

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

PO BOX 5050
BRENTWOOD TN 37024-9713

expertise is everything

subscribe on-line at www.javareport.com

Yes! I want Java Report for only **$19.50**. That's half-off the regular subscription rate!

☐ bill me ☐ payment enclosed

name

title

company

address

city state zip

phone fax

e-mail address

By providing your e-mail address, you are granting Java Report permission to contact you regarding your subscription, as well as other products and services of SIGS/101communications and/or outside parties. Please indicate those you do NOT wish to contact you.

☐ Java Report ☐ Other SIGS/101communications products ☐ Outside parties

New subscriber Discount! Save 50% if you act now!

Java™ REPORT

Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in US dollars plus shipping. Canada & Mexico $64/year, all other countries $88/year. Canadian GST included.

B91210

Yes! I want Java Report for only **$19.50**. That's half-off the regular subscription rate!

☐ bill me ☐ payment enclosed

name

title

company

address

city state zip

phone fax

e-mail address

By providing your e-mail address, you are granting Java Report permission to contact you regarding your subscription, as well as other products and services of SIGS/101communications and/or outside parties. Please indicate those you do NOT wish to contact you.

☐ Java Report ☐ Other SIGS/101communications products ☐ Outside parties

New subscriber Discount! Save 50% if you act now!

Java™ REPORT

Please allow 4-6 weeks for delivery of first issue. International subscriptions must be payable in US dollars plus shipping. Canada & Mexico $64/year, all other countries $88/year. Canadian GST included.

B91210

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 2 BRENTWOOD TN

POSTAGE WILL BE PAID BY ADDRESSEE

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

PO BOX 5050
BRENTWOOD TN 37024-9713

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 2 BRENTWOOD TN

POSTAGE WILL BE PAID BY ADDRESSEE

Java™ REPORT
THE INDEPENDENT SOURCE FOR JAVA DEVELOPMENT • www.javareport.com

PO BOX 5050
BRENTWOOD TN 37024-9713